

THESES

présentées à

LA FACULTE DES SCIENCES DE PARIS

pour obtenir

le GRADE de DOCTEUR ES SCIENCES MATHÉMATIQUES APPLIQUÉES

par

Monsieur CHEVANCE René

Sujet de la première Thèse :

« Un langage complet et cohérent pour l'informatique de gestion »

Sujet de la deuxième Thèse (propositions données par la Faculté)

« Organisation d'un ordinateur de gestion ».

Soutenues le 8 Avril 1970 devant la Commission
composée de :

Président : ARSAC

Examineurs : HOCQUENGHEM
LAUDET
NOLIN

Invité :



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41550208_0025

Que Monsieur le Professeur ARSAC et Messieurs les Professeurs du DEA d'Informatique Pratique trouvent ici l'expression de mes remerciements pour l'enseignement qu'ils m'ont dispensé.

Je tiens à remercier très sincèrement Monsieur NOLIN pour la bienveillante attention avec laquelle il a contribué au développement et à l'aboutissement du présent travail.

Je tiens à exprimer ma reconnaissance à la Direction et au personnel de l'IRIA pour l'aide qu'ils m'ont apportée.

Mes remerciements vont également à la Direction de la CII qui a bien voulu me confier cette étude dans le cadre de la Convention de Recherche de la DGRST N° 68-01-130 et qui n'a cessé de me prodiguer ses encouragements.

Je remercie enfin, tout particulièrement, les stagiaires qui m'ont été confiés par la Direction de l'IRIA.

Madame MEUNIER
Messieurs BAUDRY
GAUDICHAU
SCHNEER-SOHN

pour leur aide précieuse et la bonne humeur qu'ils ont su faire régner dans notre équipe.



1 – INTRODUCTION	1
2 – DEFINITION DU LANGAGE	3
2.1 - Généralités	3
2.1.1 - Méthode de définition syntaxique	3
2.1.2 - Alphabet du langage	4
2.1.3 - Phrases du langage	6
2.1.4 - Format des phrases du langage	7
2.1.5 - Traitement immédiat et procédures	8
2.1.6 - Macro-instructions	9
2.1.7 - Objets traités	10
2.1.7.1 - Objets simples	10
2.1.7.1.1 - Objets simples fondamentaux	10
2.1.7.1.2 - Objets simples non fondamentaux	12
2.1.7.2 - Objets groupés	13
2.1.7.2.1 - Articles	13
2.1.7.2.2 - Files	13
2.2 - Désignation des objets traités	14
2.2.1 - Constantes	14
2.2.1.1 - Constantes de mots	14
2.2.1.2 - Constantes de nombres naturels	14
2.2.1.3 - Constantes de nombres entiers	15
2.2.1.4 - Constantes d'éléments booléens	15
2.2.1.5 - Constantes de nombres décimaux	15
2.2.1.6 - Constantes de nombres réels	16
2.2.1.7 - Constantes de procédures	16
2.2.2 - Variables	17
2.3 - Descriptions	18
2.3.1 - Statut des objets traités	18
2.3.2 - Supports de l'information	18
2.3.3 - Objets simples fondamentaux	21
2.3.3.1 - Booléen — entier — naturel	21
2.3.3.2 - Mot	22
2.3.3.3 - Décimal réel	23
2.3.4 - Objets groupés	27
2.3.4.1 - Article	27
2.3.4.2 - File	28
2.3.5 - Objets simples non fondamentaux	31
2.3.5.1 - Index	31
2.3.5.2 - Procédure	32
2.3.5.3 - Ordre	32
2.3.5.4 - Etiquette	35
2.3.5.5 - Partition	35
2.3.6 - Règles générales concernant les descriptions	36

2.4 - Partitions	37
2.4.1 - Concept de partition	37
2.4.2 - Partitions indépendantes	38
2.4.2.1 - Eléments inconditionnels	38
2.4.2.1.1 - Élément simple — Format 1	39
2.4.2.1.2 - Élément répété 1ère forme — Format 2	39
2.4.2.1.3 - Élément répété 2ème forme — Format 3	46
2.4.2.1.4 - Élément doublement répété — Format 4	47
2.4.2.2 - Partitions séquentielles	48
2.4.2.3 - Eléments conditionnels	48
2.4.2.3.1 - Définition	48
2.4.2.3.2 - Type des conditions	49
2.4.2.3.2.1 - Condition du type arithmétique	49
2.4.2.3.2.2 - Condition du type booléen	50
2.4.2.3.3 - Règles concernant les descriptions d'éléments conditionnels	50
2.4.2.4 - Eléments combinés	52
2.4.2.5 - Partitions conditionnelles	53
2.4.2.6 - Conditions générales concernant la description des partitions	54
2.4.2.6.1 - Eléments répétés 1ère forme	55
2.4.2.6.2 - Eléments répétés 2ème forme	56
2.4.2.6.3 - Eléments doublement répétés	59
2.4.2.6.4 - Eléments combinés	59
2.4.3 - Cadrage des articles et des files placés en mémoire	59
2.4.4 - Partitions décrites en parallèle	63
2.4.4.1 - Eléments répétés 1ère forme	66
2.4.4.2 - Eléments répétés 2ème forme	67
2.4.4.3 - Eléments doublement répétés	68
2.4.4.4 - Eléments conditionnels	68
2.4.4.5 - Eléments combinés	68
2.4.4.6 - Format des phrases de type 4	68
2.4.5 - Sous-partition	70
2.4.6 - Partitions cataloguées	73
2.5 - Désignation des objets	74
2.5.1 - Désignation des éléments dans les partitions	74
2.5.1.1 - Élément simple	74
2.5.1.2 - Élément répété 1ère forme	75
2.5.1.3 - Élément répété 2ème forme	76
2.5.1.4 - Élément doublement répété	78
2.5.1.5 - Élément conditionnel	80
2.5.1.6 - Élément combiné	82
2.5.1.6.1 - Élément conditionnel + répétition de 1ère forme	82
2.5.1.6.2 - Élément conditionnel + répétition de 2ème forme	84
2.5.1.6.3 - Élément conditionnel + double répétition	85
2.5.2 - Généralisation dans le cas des partitions	89
2.5.3 - Emploi des expressions qualifiées	93
2.5.3.1 - Élément ou partie d'un article placé en mémoire	94
2.5.3.2 - Élément ou partie d'une file placée en mémoire	94
2.5.3.2.1 - Élément de file	94
2.5.3.2.2 - Partie de file	95
2.5.3.2.3 - Ensemble d'éléments d'une file	97
2.5.4 - Simplification des expressions qualifiées	100
2.6 - Définitions	101
2.6.1 - Objets pouvant être définis	101

2.6.2 - Objets devant être définis	102
2.6.3 - Introduction des définitions dans les programmes	103
2.7 - Termes	108
2.7.1 - Usage des termes — Forme générale	108
2.7.2 - Termes désignant des nombres naturels	112
2.7.3 - Termes désignant des nombres entiers	115
2.7.4 - Termes désignant des nombres décimaux	119
2.7.5 - Termes désignant des nombres réels	122
2.7.6 - Termes désignant des mots	124
2.7.7 - Termes et pseudo-termes désignant les positions d'un index	126
2.7.8 - Termes désignant des nombres booléens	128
2.7.9 - Termes désignant des conditions	131
2.7.10 - Termes de longueur	135
2.7.10.1 - ORD et PRE	135
2.7.10.2 - LONG	136
2.7.10.2.1 - Files	136
2.7.10.2.2 - Élément répété 2ème forme	138
2.7.10.2.3 - Élément doublement répété	138
2.8 - Instructions	139
2.8.1 - Instructions d'attribution dans une procédure	140
2.8.1.1 - Récapitulation	148
2.8.2 - Instructions de saut	149
2.8.2.1 - Instructions de saut inconditionnel	149
2.8.2.2 - Instructions de saut conditionnel	151
2.8.3 - Instructions de conversion	151
2.8.4 - Conversions implicites	161
2.8.5 - Appels de procédure	162
2.9 - Instructions de changement de support	165
2.9.1 - Introduction — Le concept de macro-instruction	165
2.9.2 - Organisation des fichiers	173
2.9.2.1 - Organisation séquentielle	173
2.9.2.2 - Organisation séquentielle indexée	174
2.9.2.3 - Organisation directe	175
2.9.2.4 - Organisation partitionnée	176
2.9.3 - Techniques d'accès	177
2.9.3.1 - Queued access technique — Technique d'accès assistée	177
2.9.3.2 - Basic access technique — Technique d'accès non assistée	177
2.9.4 - Identification des fichiers	178
2.9.4.1 - Volumes en accès direct	179
2.9.4.2 - Bandes magnétiques	180
2.9.5 - Formats des articles des fichiers	180
2.9.5.1 - Articles de longueur constante	181
2.9.5.2 - Articles de longueur variable	181
2.9.5.3 - Articles de longueur indéfinie	182
2.9.5.4 - Caractère de contrôle	182
2.9.6 - Interface avec le système d'exploitation	183
2.9.7 - Techniques de traitement des données	186
2.9.7.1 - Queued access technique — Technique d'accès assistée	186
2.9.7.1.1 - GET — Obtention d'un article	186
2.9.7.1.2 - PUT — Ecriture d'un article	187
2.9.7.1.3 - PUTX — Ecriture d'un article mis à jour	187
2.9.7.2 - Basic access technique — Technique d'accès non assistée	187
2.9.7.2.1 - READ — Lecture d'un bloc	188

2.9.7.2.2 - WRITE — Ecriture d'un bloc	188
2.9.7.2.3 - CHECK — Test de la fin d'une opération Read/Write	189
2.9.7.2.4 - WAIT — Attente de la fin d'une opération Read/Write	189
2.9.7.2.5 - Data event control block (DECB)	189
2.9.8 - Traitement des erreurs	190
2.9.8.1 - SYNADAF — Analyse des conditions SYNAD	190
2.9.8.2 - SYNADRLS — Retour d'un programme SYNAD	190
2.9.9 - Choix d'une méthode d'accès	190
2.9.10 - Ouverture et fermeture des fichiers	191
2.9.10.1 - OPEN — Ouverture d'un fichier	191
2.9.10.2 - CLOSE — Fermeture d'un fichier	191
2.9.10.2.1 - Traitement de la fin de volume	192
2.9.10.2.2 - FEØV — Création d'une fin de volume	192
2.9.11 - Acquisition et contrôle des buffers	192
2.9.11.1 - Construction d'un pool de buffers	193
2.9.11.1.1 - BUILD — Construction d'un pool de buffers	193
2.9.11.1.2 - GETPØØL — Génération d'un pool de buffers	194
2.9.11.1.3 - Construction automatique d'un pool de buffers	194
2.9.11.1.4 - FREEPØØL — Libération d'un pool de buffers	194
2.9.11.2 - Contrôle des buffers	194
2.9.11.2.1 - "Bufférisation simple"	195
2.9.11.2.2 - "Bufférisation" avec échange	196
2.9.11.3 - RELSE — Abandon d'un buffer d'entrée	198
2.9.11.4 - TRUNC — Troncation d'un buffer de sortie	198
2.9.11.5 - GETBUF — Obtention d'un buffer appartenant à un pool	198
2.9.11.6 - FREEBUF — Restitution d'un buffer à un pool	198
2.9.11.7 - FREEDBUF — Restitution d'un buffer "dynamique" à un pool	198
2.9.12 - Adaptation au langage	200
2.9.12.1 - Macro-instruction DCB	203
2.9.12.2 - Ouverture et fermeture des fichiers	203
2.9.12.2.1 - Macro-instruction OPEN	203
2.9.12.2.2 - Macro-instruction CLOSE	204
2.9.12.2.2.1 - Macro-instruction FEØV	204
2.9.12.3 - Acquisition et contrôle des buffers	204
2.9.12.3.1 - Macro-instruction BUILD	204
2.9.12.3.2 - Macro-instruction GETPØØL	204
2.9.12.4 - Traitement des erreurs	205
2.9.12.4.1 - Macro-instruction SYNADAF	205
2.9.12.5 - Traitement des fichiers séquentiels	205
2.9.12.5.1 - Queued access technique — QSAM	205
2.9.12.5.2 - Basic access technique — BSAM	205
2.9.12.5.3 - Macro-instructions communes	206
2.9.12.6 - Traitement des fichiers séquentiels indexés	206
2.9.12.6.1 - Queued access technique — QISAM	206
2.9.12.6.2 - Basic access technique — BISAM	206
2.9.12.7 - Traitement des fichiers en accès direct	206
2.9.12.8 - Traitement des fichiers partitionnés	207
2.9.12.9 - Communications avec l'opérateur	207
2.10 - Macro-instructions du compilateur	208
2.10.1 - Macro-instructions de compilation	208
2.10.1.1 - Insertion	208
2.10.1.2 - Mise en parallèle des partitions	210
2.10.1.3 - Catalogage des partitions	211

2.10.1.4 - Entrées-sorties standard	212
2.10.1.5 - Points de reprise	213
2.10.2 - Macro-instructions d'aide à la mise au point	213
2.10.2.1 - TRACE	213
2.10.2.2 - SNAP	214
2.10.3 - Macro-instructions définies par l'utilisateur	216
2.10.4 - Services du système d'exploitation	216
2.11 - Traitement immédiat et procédure	218
2.12 - Conclusion	220
3 – COMPILATEUR	223
3.1 - Introduction	223
3.2 - Phase d'analyse syntaxique	229
3.2.1 - Formalisme adopté pour la description syntaxique	229
3.2.2 - Signification des instructions	232
3.2.3 - Définition des symboles successeurs ($\mathcal{U}_v$) et alternant ($\mathcal{U}_f$) d'un symbole $\mathcal{U}_k$ donné	236
3.2.4 - Introducteur de syntaxe	238
3.2.5 - Analyseur	244
3.2.6 - Technique de "Hashing"	247
3.2.6.1 - Constitution de HDICØ	248
3.2.6.2 - Algorithme de "Hashing"	250
3.3 - Phase de vérification et de pré-génération	251
3.4 - Phase de génération	259
3.5 - Conclusion — Extension conversationnelle	260
4 – CONCLUSION	263
BIBLIOGRAPHIE	265
ANNEXES	
Annexe 1 — Version anglaise du langage	
Annexe 2 — Grammaire du langage	

1 – INTRODUCTION

Le langage présenté dans ce document est une version évoluée du langage ATF-Gestion, langage plus particulièrement destiné aux applications de gestion.

L. NOLIN a dirigé les travaux d'une équipe chargée de la définition du langage ATF-Gestion (22), en s'inspirant des principes et des idées qu'il a introduits dans le langage ATF (21).

Le présent travail a été entrepris dans le cadre de la convention de Recherche n° 68-01-130 de la Délégation Générale à la Recherche Scientifique et Technique (DGRST). Il a consisté, d'une part, à modifier sensiblement le langage ATF-Gestion, en respectant toutefois les idées qui l'ont inspiré, et, d'autre part, à définir l'architecture d'un compilateur et à en débiter la réalisation.

Tout au long de cette étude, les choix ont été faits avec un souci constant de définir logiquement le langage et en attachant une grande importance aux contraintes imposées par le domaine particulier auquel le langage doit s'appliquer.

La machine retenue, pour cette étude, est l'IBM 360 fonctionnant sous le contrôle de l'Operating System (ØS) avec MFT (Multiprogramming with a Fixed number of Tasks) ou MVT (Multiprogramming with a Variable number of Tasks). Les raisons qui ont motivé le choix de cette machine sont évoquées au paragraphe 3.1.

Le langage ATF-Gestion, résultat des travaux de l'équipe dirigée par L. NOLIN possédait certaines qualités ; parmi celles-ci, nous allons évoquer les plus significatives.

Le langage avait une grande parenté avec les langages machine :

- les descriptions étaient l'analogue des réservations,
- les définitions permettaient d'attribuer une valeur constante à certains objets du traitement,
- les instructions étaient de forme simple.

La notion de programme principal, telle qu'elle est définie dans les langages habituels, disparaissait au profit d'un programme de type particulier, le traitement immédiat, destiné à remplacer le langage des cartes-contrôle du système d'exploitation.

Les procédures correspondaient à des traitements bien définis.

Son plus grand avantage était certainement la structure logique qu'il conférait aux programmes.

Mais certains points d'importance avaient été laissés dans l'ombre.

La notion de partition n'était pas clairement dégagée.

Les instructions d'entrée-sortie ne se présentaient pas sous une forme très réaliste

Les relations avec le système n'étaient pas traitées.

Et surtout, le compilateur, expression ultime de la syntaxe et de la sémantique n'était pas réalisé.

Les principales évolutions concernent :

- la présentation plus stricte du langage.
- la précision et l'extension importante de la notion de partition,
- le traitement original des entrées-sorties par l'introduction, dans le langage même, de procédures incomplètes - au sens ATF - que nous avons dénommées macro-instructions.

De cette façon, les qualités du langage ont été conservées, ses défauts supprimés et sa portée a été considérablement étendue.

Cette façon de procéder pour la définition et le développement d'un langage qui dérive directement d'un langage complet logiquement défini, l'ATF (21), laisse à penser que d'autres langages peuvent se déduire de cette manière.

En conséquence, la désignation du langage a été changée en :

CØDELAB

(abréviation de "CØherantly DEvelopped LAnguages-Business")

La première partie de ce document concerne la définition du langage. La seconde partie décrit la structure du compilateur ainsi que la mise en oeuvre d'une méthode d'analyse syntaxique descendante.

2 — DEFINITION DU LANGAGE

2.1 - GENERALITES

2.1.1 - Méthode de définition syntaxique

Avant de commencer la description du langage, il faut choisir une méthode de définition syntaxique

La forme de BACKUS permet de décrire avec précision la syntaxe du langage, mais elle n'est pas très imagée.

Le langage utilisé dans la description du CØBØL, au contraire, "parle mieux" aux usagers ; aussi est-ce un langage inspiré de ce dernier qu'on utilise dans cette première partie de l'étude.

Pour les besoins de l'analyse syntaxique, un langage particulier, dérivé de la forme de BACKUS, a été défini (3.2.1). L'Annexe 2 donne la grammaire du langage exprimée sous cette forme. Elle est destinée aux spécialistes de la compilation

Les règles du langage de description (en fait du méta-langage) sont :

1. Tous les mots en lettres capitales doivent figurer dans les programmes, lorsque la partie de phrase, dont ils font partie, est requise.
2. Les mots entre < et > représentent l'utilisation de symboles, dont le type est défini par le nom figurant entre < et >, fournis par l'utilisateur.
3. Les termes entre { et } indiquent qu'un choix doit être fait parmi le contenu.
4. Les termes entre } et { suivis de '...' indiquent une répétition des termes entre crochets.
5. Les termes compris entre [et] représentent une option, les termes compris entre ces crochets pouvant être présents ou non au gré de l'utilisateur.

2.1.2 - Alphabet du langage

Les mots du langage sont formés à partir des **symboles** de l'alphabet suivant

- l'espace ou le blanc	□
- les lettres majuscules	ABCDEFGHIJKLMNOPQRSTUVWXYZ
- les chiffres	0123456789
- les signes d'opération	+ - * / **
- les symboles de comparaison	= < > ≤ ≥ ≠
- le symbole d'affectation	:=
- les parenthèses	()
- les guillemets de citation.	' "
- les signes de ponctuation	. , : ;
- la barre de séparation	
- le symbole monétaire	F ou \$
- les symboles spéciaux	? @ ! %
- le symbole séparateur	&

Les ordinateurs envisagés dans l'application utilisent le code à 8 bits EBCDIC (Extended Binary Coded Decimal Interchange Code); en conséquence certains symboles de l'alphabet sont représentés par un caractère différent ou composés de 2 caractères EBCDIC. Ce sont :

- le symbole d'exponentiation ** formé de 2 caractères *
- le symbole de comparaison ≤ (inférieur ou égal à) formé des deux caractères < =
- le symbole de comparaison ≥ (supérieur ou égal à) formé des deux caractères > =
- le symbole de comparaison ≠ représenté par le caractère #
- le symbole d'affectation := représenté par les deux caractères : suivi de =

Les mots du langage sont groupés, suivant les règles exposées ci-après, pour former des **phases**.

Certains des mots qu'il est possible de former sont réservés à un usage spécial.

Ce sont les **mots-clés** ou **mots-réservés**, ci-après :

ABS	ERREUR	NØN
ADLIB	ERUBAN	ØRD
AINE	ET	ØRDRE
APRES	ETIQUETTE	ØU
AR	FAUX	PAPIER
ARRØNDIA	FILE	PARTITIØN
ARTICLE	FIN	PRE
AUXILIAIRE	G (G	PREMIER
AV	G) G	PRØCEDURE
AVANT	IEME	PRØCHE
BANDE	IMPRIME	PUIS
BLANC	INDEX	REEL
BØØLEEN	INFER	RESTE
CATAL	JUSQUA	RESULTAT
CLAVIER	LIMITEA	RIEN
DE	LØNG	SCARTE
DECIMAL	MACRØ	SI
DERNIER	MEMØIRE	SIGNE
DØNNEE	MIXTE	SRUBAN
ECARTE	MØD	STØCK
EFFACER	MØT	SUPER
ENTIER	NATUREL	SYNCHRØ

TRAITEMENTIMMEDIAT	VRAI
UNITE	ZERØ
VERS	
VIDE	

La liste des mots réservés de la version anglaise du langage est donnée à l'Annexe 1.

2.1.3 - Phrases du langage

Un programme – traitement immédiat ou procédure – est composé des phrases de 8 types différents, dans l'ordre :

1. Type 0 - Début

Une phrase du type 0 doit figurer au début de tout programme, elle contient un code (le chiffre 0) et le nom du programme.

2. Type 1 - Références

Les phrases de type 1 permettent de citer les procédures auxquelles le programme en cours de description fait référence.

3. Type 2 - Descriptions

Les phrases de type 2 permettent de décrire les objets du traitement.

4. Type 3 - Partitions

Les phrases de type 3 permettent de décrire la structure de certains objets.

5. Type 4 - Changement de support

Les phrases de type 4 permettent de mettre en parallèle les structures de 2 objets, et partant de définir les transformations que subit un objet au cours d'une opération de changement de support ou au cours de certaines instructions d'affectation.

6. Type 5 - Définitions

Les phrases de type 5 permettent d'affecter des valeurs à certains objets qui, de ce fait, deviennent des constantes vis à vis du programme en cours de description.

7. Type 6 - Traitement

Les phrases de type 6 permettent de décrire les instructions à exécuter.

8. Type 7 - Fin

Une phrase de type 7 doit figurer à la fin de tout programme; elle contient un code (le chiffre 7) et le nom du programme.

En résumé, un programme est composé dans l'ordre de :

- une phrase de type 0, puis
- zéro, une ou plusieurs phrases de type 1, puis
- zéro, une ou plusieurs phrases de type 2, puis
- zéro, une ou plusieurs phrases de type 3, puis
- zéro, une ou plusieurs phrases de type 4, puis
- zéro, une ou plusieurs phrases de type 5, puis
- une ou plusieurs phrases de type 6, puis
- une phrase de type 7.

2.1.4 - Format des phrases du langage

Dans la première version du langage, il avait été introduit 6 types de feuilles de programmation, ces feuilles comportant des zones. Les 6 types de feuilles servant à l'écriture, respectivement, des 6 types de phrases (1 à 6) du langage ; les phrases de type 0 et 7 étant élémentaires et peu utilisées ne justifiaient pas la définition d'un 7ème type de feuille.

Le fait de fixer le format des phrases introduisait d'assez grandes contraintes de programmation (e.g. longueur des identificateurs).

Il a semblé judicieux d'introduire un format libre, moyennant l'usage d'un séparateur de zones, ce qui augmente la souplesse du langage, bien que l'usage en soit moins simple.

Généralement, un programme est présenté sur cartes perforées, ou images de cartes perforées s'il a été introduit sur un support magnétique (e.g. disque, bande, tambour).

Pour le langage, la partie utilisable d'une carte perforée s'étend de la colonne 1 à la colonne 72 comprise. Les colonnes allant de 73 à 80 sont réservées à une numérotation ou identification éventuelle.

La colonne 1 joue un rôle spécial, la présence d'un 'C' dans cette colonne indique une carte commentaire.

Comme il est indiqué ensuite, la plupart des phrases se décomposent en zones. Avec le format fixe de la 1ère version du langage, les zones étaient associées à des numéros de colonnes. En format libre, le passage d'une zone à la suivante est indiqué à l'aide du séparateur ' & '.

Une phrase se termine toujours par un point virgule: ' ; '.

La succession de plusieurs séparateurs ' & ' indique l'absence des zones correspondantes.

Il est possible d'abrégier l'écriture d'une phrase par un point virgule ' ; ' sans citer les séparateurs relatifs aux zones suivantes dans le cas où celles-ci sont vides.

Une phrase peut s'étendre sur plusieurs cartes.

Les cartes commentaires peuvent être placées à n'importe quel endroit dans les cartes du programme.

Les blancs ou espaces sont facultatifs (sauf indication contraire) et peuvent figurer en nombre quelconque.

2.1.5 - Traitement immédiat et procédures

Un traitement se compose d'un programme particulier le TRAITEMENT IMMEDIAT et éventuellement d'une ou plusieurs procédures.

Le traitement immédiat est l'homologue du PRØGRAMME en langage ATF (21).

Il a pour but de faire passer une certaine partie du langage de commande du système de l'ordinateur dans le langage de programmation.

Par rapport à une procédure, un traitement immédiat se caractérise par les points suivants :

- il est impossible dans un quelconque autre programme (i.e. traitement immédiat ou procédure) de faire référence à un traitement immédiat. Celui-ci s'apparente, à ce point de vue, à la notion classique de programme principal.
 - les descriptions des objets du traitement servent, outre le fait descriptif, à réserver les ressources nécessaires.
- Ceci est équivalent aux ordres du langage de commande du système relatifs à l'allocation de ressources pour une tâche.

- les instructions qu'il est possible d'y faire figurer sont de type particulier, sous-ensemble des instructions qui peuvent figurer dans une procédure.

Cette question est reprise en détail au paragraphe 2.11.

La définition des procédures du langage est identique à celle des procédures complètes de l'ATF (21) ; les procédures incomplètes n'ont pas été retenues du fait de leur complexité eu égard au peu d'usage qui peut en être fait dans le traitement des problèmes de gestion.

Les procédures se présentent comme des instructions, en général complexes, qu'il suffit de référencer pour les faire exécuter.

La répartition des arguments en données, mixtes et résultats et les types d'arguments autorisés doivent faciliter la conception, la compréhension et la mise au point des programmes.

Dans une procédure, toutes les instructions sont permises y compris l'appel de cette même procédure (récurrence).

La souplesse et la puissance des procédures est légèrement contrebalancée par une augmentation du temps d'exécution due à la transmission des paramètres et à la génération de certaines zones.

Une possibilité a été introduite, qui consiste à insérer la procédure dans un programme à la place de son appel.

Cette technique permet d'avoir à la fois une écriture plus facile d'un traitement et une mise au point moins fastidieuse, du fait du découpage en procédures, et un programme résultant efficace après insertion de procédures.

2.1.6 - Macro-instructions

Il est apparu, lors de l'étude de la présente version du langage, que la notion de procédure, très logiquement définie en ATF (21), demandait à être complétée pour réaliser certains traitements.

En même temps certaines imprécisions de la première version nécessitaient des modifications.

Tout ceci a permis de dégager le concept de macro-instruction.

Dans un appel de procédure, ainsi qu'on le verra par la suite, seuls peuvent figurer comme arguments des objets du traitement et non des étiquettes, ce qui est logique puisque les étiquettes sont des éléments internes à un programme. Ceci est utile pour les entrées-sorties.

De même certaines facilités : insertion de procédures, aide à la mise au point, etc..., ne peuvent s'exprimer par des appels de procédures.

Les macro-instructions se divisent en 4 catégories :

- les macro-instructions du compilateur, définies à l'implantation du compilateur ;
- les macro-instructions d'entrée-sortie, liées au triplet : ordinateur, système d'exploitation, compilateur ;
- les macro-instructions utilisées pour les services du système d'exploitation ;
- les macro-instructions définies par l'utilisateur.

La référence à une macro-instruction comporte deux parties; la première est de forme identique à un appel de procédure, i.e. nom de la macro-instruction (débutant par MACRØ) et séquence d'appel; la seconde partie comporte une suite de doublets : caractéristique, spécification.

2.1.7 - Objets traités

Le langage permet de traiter des objets **simples** et des objets **groupés**.

2.1.7.1 - OBJETS SIMPLES

Les objets simples se divisent en deux grandes catégories :

- les objets simples fondamentaux
- les objets simples non fondamentaux.

2.1.7.1.1 - OBJETS SIMPLES FONDAMENTAUX

Les objets simples fondamentaux sont les objets sur lesquels il est possible de faire des opérations simples.

Parmi ces objets, on distingue :

Les mots du langage

ce sont les séquences des symboles de l'alphabet du langage, e.g.

GAMMA

$A + B - C * D$

10% PØURQUØI ?

La longueur, i.e. le nombre de caractères (et non de symboles) de ces mots est limité à 256.

Les nombres naturels

ce sont des nombres entiers positifs ou nuls (nombres qui appartiennent à $\mathbb{N}$)

Le plus grand nombre naturel exprimable est $2^{31}-1$ soit 2 147 483 647

Cette limitation est due à la représentation sur 32 bits permise par l'ordinateur utilisé.

Les nombres entiers

ce sont les nombres entiers positifs, nuls ou négatifs (nombres qui appartiennent à $\mathbb{Z}$)

Le plus grand nombre entier est limité en module à $2^{31}-1$ soit 2 147 483 647.

Cette limitation, comme pour les nombres naturels, est due aux possibilités "hardware".

Les nombres booléens

ce sont des séquences d'éléments 'VRAI' ou 'FAUX'.

La longueur des nombres booléens, i.e. le nombre d'éléments booléens dans un nombre, est limité à 2048 (256 fois 8 éléments).

Il est possible de définir un nombre booléen de deux façons :

1. une séquence des mots VRAI et FAUX ;

2. une séquence des chiffres 0 et 1 ,

0 étant équivalent à FAUX et 1 à VRAI.

Cette double possibilité est envisagée pour des commodités d'utilisation.

Les nombres décimaux

Dans la première version du langage, la notion de "décimal" demandait à être précisée ; aussi a-t-on introduit deux catégories de nombres :

– les nombres décimaux ;

– les nombres réels.

Les nombres décimaux sont représentés en machine (e.g. IBM 360 et CII 10070) sous forme de deux caractères décimaux par octet. La position du point décimal étant une extension "software".

Ces nombres obéissent à des règles arithmétiques particulières (2-7-4).

La longueur d'un nombre décimal, i.e. le nombre de chiffres décimaux dans le nombre, est limité à 31 chiffres, plus signe.

Les nombres réels

ce sont les nombres appelés habituellement "nombres en virgule flottante". Ils sont représentés en format semi-logarithmique : mantisse et exposant, et sont une approximation des nombres appartenant à $\mathbb{R}$.

Ils obéissent à des règles arithmétiques différentes de celles des nombres décimaux (2-7-5).

Deux formats sont possibles :

- les nombres réels simple précision ;
- les nombres réels double précision,

en fonction du nombre de chiffres significatifs de la mantisse.

La valeur absolue de l'exposant d'un nombre réel est limitée à 78.

Le nombre de chiffres de la mantisse est limité à :

- en simple précision à 7 chiffres,
- en double précision à 17 chiffres.

L'ensemble des objets simples fondamentaux est désigné par le mot réservé ADLIB (ad libitum).

2.1.7.1.2 - OBJETS SIMPLES NON FONDAMENTAUX

Les autres objets simples sont :

- les index de files
- les procédures
- les étiquettes
- les partitions.

La signification de ces objets est explicitée plus bas.

2.1.7.2 - OBJETS GROUPES

Les objets groupés sont des ensembles d'objets simples (fondamentaux ou non).

Il existe deux sortes d'objets groupés :

- les articles
- les files

2.1.7.2.1 - ARTICLES

Un article est un groupement d'objets simples de natures différentes en général. Un article est structuré par une **partition** (2.4).

2.1.7.2.2 - FILES

Des articles **structurés par une même partition** peuvent être groupés dans une file.

Pour traiter une file, un certain nombre d'index lui sont attachés : ils ont pour rôle de repérer à un instant donné du traitement un article et un seul de cette file. Les index peuvent être placés dans une position déterminée et déplacés à volonté au cours du traitement.

En fonction du nombre d'entrées, on distingue :

- les fichiers, ils sont composés d'une suite simple d'articles : ce sont des files à une entrée. (Pour les imprimantes, un fichier peut être une file à deux entrées).
- les files en général, qui possèdent n entrées (où n est un nombre naturel quelconque au moins égal à 1)

A chaque entrée sont attachés un certain nombre d'index et un élément de la file est repéré par une suite ordonnée d'index dont le premier est attaché à la première entrée, le deuxième à la deuxième, etc...

2.2 - DESIGNATION DES OBJETS TRAITES

2.2.1 - Constantes

Certains objets sont connus et ne subissent aucun changement durant le traitement. Ils sont désignés par des constantes, mots particuliers du langage.

2.2.1.1 - CONSTANTES DE MOTS

Un mot qui ne contient pas de guillemets est désigné par ce mot lui-même précédé du guillemet ouvrant ' et suivi du guillemet fermant " .

Le nombre de caractères figurants dans une constante de mot, non compris les guillemets, est limité à 256.

Pour obtenir la constante qui désigne un mot dans lequel apparaissent des guillemets, tout guillemet ouvrant est remplacé par G (G et tout guillemet fermant par G) G. Le mot ainsi obtenu est placé entre un guillemet ouvrant et un guillemet fermant ; e.g.

$C\emptyset N\emptyset T\emptyset N\emptyset T\emptyset D\emptyset M\emptyset T = 'SEQUENCE\emptyset DE\emptyset CARACTERES'$

est désigné par :

$C\emptyset N\emptyset T\emptyset N\emptyset T\emptyset D\emptyset M\emptyset T = G(G\ SEQUENCE\emptyset DE\emptyset CARACTERES\ G)G$

Tous les blancs ($\emptyset$) compris entre les guillemets sont inclus dans la constante de mot.

REMARQUE :

G (G et G) G ne comptent que pour 1 caractère chacun. (Ils citent ' et " respectivement).

2.2.1.2 - CONSTANTES DE NOMBRES NATURELS

Un nombre naturel est désigné par une suite de chiffres, e.g. :

0 désigne le nombre zéro

21 désigne le nombre vingt et un

021 désigne aussi le nombre vingt et un

On peut considérer que 21 est le "nom propre", 021 est un "pseudonyme" du nombre vingt et un.

La description d'une constante de nombre naturel ne doit pas comprendre de blancs.

La valeur maximale d'un nombre naturel est 2 147 483 647.

2.2.1.3 - CONSTANTES DE NOMBRES ENTIERS

Un nombre entier est désigné, soit par une constante de nombre naturel, soit par une telle constante précédée du signe $+$ ou du signe $-$; ainsi

plus vingt et un est désigné par 21 ou $+ 21$

moins vingt et un est désigné par $- 21$

REMARQUES :

$+ 0$ et $- 0$ désignent 0

Le signe éventuel peut être séparé du nombre par un ou plusieurs blancs.

La valeur maximale d'un nombre entier est limitée en module à 2 147 483 647.

2.2.1.4 - CONSTANTES D'ELEMENTS BOOLEENS

Un élément booléen est désigné par VRAI ou 1 ou par FAUX ou 0.

Une constante d'élément booléen est désignée par une séquence d'éléments VRAI ou FAUX séparés par un ou plusieurs blancs ou par une séquence des chiffres 1 ou 0 non séparés par des blancs.

Le nombre maximal d'éléments booléens qu'il est possible de faire figurer dans une séquence est limité à 2048.

2.2.1.5 - CONSTANTES DE NOMBRES DECIMAUX

Un nombre décimal est désigné soit :

- par une constante d'entier
- par une constante d'entier (qui désigne la partie entière de ce nombre) suivie d'un point et éventuellement suivi lui-même d'une constante de nombre naturel (nom propre de la partie décimale), e.g. :
 - le nombre décimal $- 21,0$ s'écrit $- 21$ ou $- 021$ ou $- 21.0$ etc...
 - le nombre décimal $+ 102,06$ s'écrit $+ 102.06$ ou 102.060 etc...

Dans le cas où la partie décimale du nombre est nulle il est possible, soit d'écrire le nombre comme une constante d'entier, soit de faire figurer le point décimal après la partie entière.

REMARQUES :

+ 0.0 et - 0.0 désignent zéro.

Le nombre maximum de chiffres décimaux qu'il est possible de faire figurer dans une constante de nombre décimal (partie entière et partie décimale) est limité à 31.

2.2.1.6 - CONSTANTES DE NOMBRES REELS

Deux désignations sont possibles pour les nombres réels, d'une part une représentation analogue à celles des nombres décimaux (2-2-1-5) et d'autre part le format semi-logarithmique classique pour les nombres en virgule flottante :

$$\left[\left\{ \pm \right\} \right] < \text{nb naturel} > \left[\left[< \text{nb naturel} > \right] \right] \left[E \left\{ \begin{smallmatrix} + \\ - \\ \square \end{smallmatrix} \right\} < \text{nb naturel} > \right]$$

La valeur absolue maximale de l'exposant est limitée à 78.

Le nombre de chiffres décimaux de la mantisse (i.e. le nombre total de chiffres spécifié pour la constante) est limité à :

7 en simple précision

17 en double précision

REMARQUE :

Une constante de nombre réel dont la mantisse, quel que soit le signe qui la précède (+ ou -), est nulle (quel que soit la valeur de la caractéristique) désigne zéro.

2.2.1.7 - CONSTANTES DE PROCEDURES

Il n'existe que deux constantes de procédure :

- ERREUR qui désigne la procédure exécutée à la demande de l'utilisateur, à l'exécution du programme, quand une erreur ne permet pas de poursuivre le traitement ;
- VIDE dont l'effet est de "ne rien faire".

Aux procédures, on peut ajouter les macro-instructions mises à la disposition de l'utilisateur par le compilateur. Il y a des constantes de macro-instructions (2-9).

Il n'y a pas de constantes d'étiquettes, de partitions, de files ni d'index.

2.2.2 - Variables

Certains objets peuvent varier au cours d'un traitement.

Ils sont désignés par des **identificateurs**, des **expressions** et éventuellement des **termes**.

Les variables les plus simples sont les identificateurs. Un identificateur est un mot composé de lettres et de chiffres et commençant par une lettre, e.g.

A A10 MATRICULE1

Cet identificateur ne doit pas être un mot réservé (2-1-2).

La nature d'une variable (i.e. le type d'objet qu'elle désigne) est fixé en toute liberté au début d'un programme dans la partie déclaration.

Par rapport à la première version du langage, la possibilité de pouvoir séparer un identificateur en deux ou plusieurs parties par des blancs a été supprimée ; les risques d'erreur étant relativement grands. L'une des extensions possibles du langage est d'introduire un caractère séparateur comme en CØBØL ou en PL/1.

Le nombre maximal des caractères significatifs d'un identificateur est limité à 30.

Les autres désignations possibles des variables sont décrites plus bas.

La notion d'identificateur, définie ci-dessus, est référencée par la suite à l'aide de la notation :

< identificateur >

2.3 - DESCRIPTIONS

Les descriptions ont pour objet de décrire tous les objets traités dans un programme, à l'exception des constantes.

Les descriptions s'expriment par des phrases de type 2.

2.3.1 - Statut des objets traités

Par rapport au traitement que le programme leur impose, ces objets se répartissent en quatre catégories :

1. **les données** qui sont utilisées sans être modifiées ;
2. **les résultats** qui sont calculés sans qu'il soit jamais tenu compte de leur valeur antérieure, si tant est qu'ils en aient une ;
3. **les mixtes** qui tiennent des uns et des autres car il est possible d'utiliser leur valeur antérieure et éventuellement de remplacer celle-ci par une nouvelle valeur.

Ces trois catégories d'objets constituent les **arguments** d'une procédure. La division des arguments d'une procédure en 3 catégories supprime nombre de problèmes concernant la passation des arguments. En effet une constante ne peut figurer que dans les arguments de type donnée d'une procédure ; les instructions de cette procédure ne peuvent qu'utiliser la valeur de cette constante.

Toute utilisation d'une variable de statut "donnée" dans une instruction quelconque en une position qui entraîne une modification de la valeur de cette variable est une erreur détectée à la compilation.

Cette distinction évite toute confusion et toute modification intempestive.

4. **les auxiliaires**, qui sont des objets créés pour les besoins internes du traitement dans un programme. Aucune restriction particulière ne concerne l'emploi des auxiliaires.

Cette division en classes de l'ensemble des variables d'après leur statut est identique à celle introduite par L. NOLIN dans l'ATF de base (21) et reprise dans la première version du langage.

2.3.2 - Supports de l'information

Les objets traités sont **représentés sur des supports**.

Les différents supports considérés sont :

– MEMØIRE	mémoire centrale de l'ordinateur
– STØCK *	tout support adressable autre que MEMØIRE (e.g. disque, tambour)
– BANDE	bande magnétique
– ECARTE	cartes perforées en entrée
– ERUBAN	ruban perforé en entrée
– CLAVIER	lignes frappées sur le clavier d'une machine à écrire ou tout autre appareil similaire.
– SCARTE	cartes perforées en sortie
– SRUBAN	ruban perforé en sortie
– PAPIER	lignes sorties sur une machine à écrire ou tout autre appareil similaire.
– IMPRIME	lignes ou pages écrites sur une imprimante ou tout autre appareil similaire.

* Dans le cas de STØCK, il est nécessaire de distinguer le type d'organisation du fichier, i.e. séquentiel, séquentiel indexé, direct ou partitionné. La distinction est faite par un suffixe ajouté à STØCK ; e.g. STØCKSEQ signifie organisation séquentielle du fichier ; de plus amples détails sont donnés au paragraphe 2.9.2.

La nature des objets traités et leur support ne sont pas indépendants. Dans les supports considérés, il existe deux grandes catégories :

- les supports internes : MEMØIRE, STØCK, BANDE
- les supports externes : les autres.

Seuls des articles ou des files dont les éléments sont tous de nature MØT peuvent prendre place sur des supports externes (2-9-12).

De même le statut et le support ne sont pas indépendants, le tableau suivant indique les différentes combinaisons possibles, dans un programme du type PRØCEDURE :

STATUT	SUPPØRT
DØNNEE, MIXTE, RESULTAT, AUXILIAIRE	MEMØIRE STØCK BANDE
DØNNEE	ECARTE ERUBAN CLAVIER
RESULTAT	SCARTE SRUBAN PAPIER IMPRIME

Les combinaisons sont légèrement différentes pour un programme du type TRAITEMENT IMMEDIAT (voir 2.11).

Dans un programme, **les descriptions** servent à préciser le statut, la nature, le support et éventuellement la structure des objets traités.

Le format général des phrases de type 2, qui expriment les descriptions, est :

< statut > & < nature > & < nom > & < longueur > & < nom partition > & < place partition > &
 < numéro entrée > & < nom index > { , < nom index > } ... } & & & & & & &
 < numéro entrée > & < nom index > { , < nom index > } ... } ... & < place > ;

La signification des termes apparaissant dans les différentes zones sera précisée plus loin

2.3.3 - Objets simples fondamentaux

Les descriptions d'objets simples fondamentaux (ADLIB) ont la forme suivante :

< statut > & < nature > & < identificateur > & < longueur > & & & & MEMØIRE ;

Les différences portent sur la spécification de la longueur.

2.3.3.1 - BØØLEEN - ENTIER - NATUREL

$$\left(\begin{array}{c} DØNNEE \\ MIXTE \\ RESULTAT \\ AUXILIAIRE \end{array} \right) \& \left(\begin{array}{c} BØØLEEN \\ ENTIER \\ NATUREL \end{array} \right) \& \< \text{identificateur} \> \& \< \text{longueur} \>$$

[& [& [& [& [MEMØIRE]]]]] ;

< identificateur > est le nom de l'objet décrit,

< longueur > est une expression définissant la longueur de l'objet qui peut être :

- une constante de nombre naturel
- une variable de nature NATUREL et de statut DØNNEE ou MIXTE
- un terme de longueur (2-7-10)
- rien, dans le cas où l'élément en cours de description est de statut DØNNEE, MIXTE ou RESULTAT la longueur est fixée à l'exécution au moment de l'appel de procédure. La longueur prise est celle du paramètre de l'appel de procédure.
- rien, dans le cas où l'élément en cours de description est de statut AUXILIAIRE la valeur adoptée est alors la valeur standard.

Sauf pour les nombres booléens, les valeurs standards en cas d'absence ou d'erreur sont les valeurs maximales des longueurs :

BOOLEEN	32	(nombre d'élément booléens)
NATUREL	9	
ENTIER	9	

Les différentes longueurs possibles sont :

BOOLEEN	l'objet est représenté sur un nombre de bits égal au multiple de 8 immédiatement supérieur à tout nombre naturel depuis 1 jusqu'à 2048.
NATUREL	1, 2 l'objet est représenté sur 1 octet 3, 4 l'objet est représenté sur 2 octets 5, 6, 7, 8, 9 l'objet est représenté sur 4 octets
ENTIER	1, 2, 3, 4 l'objet est représenté sur 2 octets 5, 6, 7, 8, 9 l'objet est représenté sur 4 octets

2.3.3.2 - MOT

$\left\{ \begin{array}{l} \text{DONNEE} \\ \text{MIXTE} \\ \text{RESULTAT} \\ \text{AUXILIAIRE} \end{array} \right\}$	$\& \text{ MOT } \& \< \text{identificateur} \> \& \< \text{longueur} \> \left[\& \left[\& \left[\& \left[\& \left[\right. \right. \right. \right. \right. \right. \right.$
---	--

MEMOIRE]]]]]] ;

< identificateur > est le nom de l'objet décrit,

< longueur > est une expression qui définit le nombre de caractères qui composent cet objet.

Les valeurs possibles sont :

- une constante de nombre naturel
- une variable de nature NATUREL et de statut D~~ON~~NÉE ou MIXTE
- un terme de longueur (2-7-10).
- rien, dans le cas où l'objet en cours de description est de statut D~~ON~~NÉE, MIXTE ou RESULTAT la longueur est fixée à l'exécution au moment de l'appel de procédure.
- rien, dans le cas où l'objet en cours de description est de statut AUXILIAIRE, la valeur standard de la longueur est adoptée.

La longueur maximale d'une variable de nature M~~OT~~ est 256 caractères

2.3.3.3 - DECIMAL-REEL

Le format pour la description d'un objet de nature DECIMAL est :

$$\left\{ \begin{array}{l} D\cancel{O}NNEE \\ MIXTE \\ RESULTAT \\ AUXILIAIRE \end{array} \right\} \& \text{ DECIMAL } \& \langle \text{identificateur} \rangle \& \langle \text{longueur 1} \rangle$$

$$\left[\& \left[\& \left[\& \left[\& \left[\text{MEM}\cancel{O}IRE \right] \right] \right] \right] \right] \right] ;$$

< identificateur > est le nom désignant l'objet décrit.

< longueur 1 > est une expression définissant le nombre de chiffres décimaux de la variable.

La forme générale de < longueur 1 > est :

< longueur > . < longueur >

où < longueur > peut être :

- une constante de nombre naturel ;

- une variable de nature NATUREL et de statut DØNNEE ou MIXTE ;
- un terme de longueur (2-7-10) ;
- rien.

En conséquence l'une des formes possibles est :

$$\left(\begin{array}{l} \text{DØNNEE} \\ \text{MIXTE} \\ \text{RESULTAT} \\ \text{AUXILIAIRE} \end{array} \right) \& \text{ DECIMAL } \& \text{ < identificateur > } \&$$

$$[\& [\& [\& [\& [\text{ MEMØIRE }]]]]] ;$$

La première expression de longueur exprime le nombre maximal de chiffres décimaux avant le point décimal et la seconde le nombre maximal de chiffres décimaux après ce point décimal.

La somme de ces deux nombres ne peut être supérieure à 31.

Dans le cas où l'élément décrit est de statut DØNNEE, MIXTE ou RESULTAT et où l'une ou les deux expressions de longueur sont absentes, les longueurs sont fixées à l'exécution au moment de l'appel de procédure. Si la somme des deux longueurs excède 31, une erreur est signalée.

Lorsque l'élément décrit est de statut AUXILIAIRE deux cas peuvent se présenter :

1. l'une des deux expressions de longueur est absente :
il est attribué à l'expression de longueur manquante une valeur par défaut égale à :
 $31 - (\text{valeur de l'expression de longueur formulée})$
2. < longueur > est de la forme :
La partie entière est supposée avoir une longueur de 31 chiffres décimaux, la partie décimale ayant une longueur nulle.

Dans le cas où l'expression de longueur est omise on peut écrire, plus simplement :

$$\left\{ \begin{array}{l} \text{DONNÉE} \\ \text{MIXTE} \\ \text{RESULTAT} \\ \text{AUXILIAIRE} \end{array} \right\} \& \text{DECIMAL} \& \text{ < identificateur > } \&$$

$$[\& [\& [\& [\& [\text{MEMOIRE}]]]]] ;$$

Le format général pour la description d'un objet de nature REEL est :

$$\left\{ \begin{array}{l} \text{DONNÉE} \\ \text{MIXTE} \\ \text{RESULTAT} \\ \text{AUXILIAIRE} \end{array} \right\} \& \text{REEL} \& \text{ < identificateur > } \& \text{ < longueur 2 > }$$

$$[\& [\& [\& [\& [\text{MEMOIRE}]]]]] ;$$

< identificateur > est le nom de l'objet décrit,

< longueur 2 > est une expression qui définit le format du nombre réel. Comme on l'a mentionné plus haut, il existe deux formats pour les nombres réels (ou virgule flottante) :

- simple précision
- double précision

La forme générale de < longueur 2 > est :

$$\text{ < longueur > } , \text{ < longueur > }$$

où < longueur > peut être :

- une constante de nombre naturel ;
- une variable de nature NATUREL et de statut DONNÉE ou MIXTE ;
- un terme de longueur (2-7-10) ;
- rien

En conséquence l'une des formes possibles est :

$$\left(\begin{array}{l} \text{D}\text{ONNEE} \\ \text{MIXTE} \\ \text{RESULTAT} \\ \text{AUXILIAIRE} \end{array} \right) \& \text{ REEL } \& \langle \text{identificateur} \rangle \& ,$$

$$\left[\& \left[\& \left[\& \left[\text{MEM}\text{OIRE} \right] \right] \right] \right] ;$$

La première expression de longueur donne la limite supérieure en valeur absolue de l'exposant. A vrai dire elle n'a été conservée que pour rappeler les notations de l'ATF de base concernant l'arithmétique en longueur variable. La valeur de cette expression est sans importance, dans l'application présente, l'exposant prenant des valeurs, quel que soit le format machine du nombre réel (simple ou double précision), appartenant à $[-78, +76]$.

La seconde expression de longueur donne le nombre de chiffres décimaux significatifs * de la mantisse.

Par convention une longueur inférieure ou égale à 7 entraîne l'adaption du format virgule flottante simple précision pour la représentation de la variable

Une valeur comprise entre 8 et 17 entraîne l'adoption du format virgule flottante double précision pour la représentation de la variable.

En fait, seule la seconde expression de longueur importe.

Dans le cas où l'élément décrit est de statut **D**ONNEE, **M**IXTE ou **R**ESULTAT et où la seconde expression de longueur est absente ou fixée par une variable de nature **N**ATUREL et de statut **D**ONNEE ou **M**IXTE, le format "virgule flottante" est fixé à l'exécution au moment de l'appel de procédure.

• REMARQUE :

La notion de nombre de chiffres significatifs en virgule flottante appelle quelques commentaires, il est considéré ici comme le nombre de chiffres décimaux équivalent au nombre de bits de la mantisse, soit N ce nombre de chiffres significatifs, il est calculé à l'aide de :

$$N = (\text{Nombre de bits de la mantisse}) * \log_{10} 2$$

Un nombre virgule flottante simple précision possède une mantisse de 24 bits et un nombre en double précision une mantisse de 56 bits.

Dans le cas où l'élément décrit est de statut AUXILIAIRE deux cas particuliers se présentent :

- la seconde expression de longueur est fixée par une variable de nature NATUREL et de statut DØNNEE ou MIXTE ; alors le format virgule flottante adopté est fixé à l'exécution au moment de l'appel de procédure.
- l'expression de longueur est absente ; alors le format virgule flottante simple précision, option standard, est adopté .

Dans le cas où l'expression de longueur est omise on peut aussi décrire un REEL sous la forme suivante :

$$\left(\begin{array}{l} DØNNEE \\ MIXTE \\ RESULTAT \\ AUXILIAIRE \end{array} \right) \& \text{REEL} \& \text{ < identificateur > } \& \\ \left[\& \left[\& \left[\& \left[\& \left[\text{MEMOIRE} \right] \right] \right] \right] \right] \right]$$

2.3.4 - Objets groupés

2.3.4.1 - ARTICLE

Le format général pour la description d'un article est :

$$\left(\begin{array}{l} DØNNEE \\ MIXTE \\ RESULTAT \\ AUXILIAIRE \end{array} \right) \& \text{ARTICLE} \& \text{ < identificateur 1 > } \& \& \text{ < identificateur 2 > } \\ \& \left[\text{CATAL} \right] \left[\& \left[\& \left[\text{ < place > } \right] \right] \right] ;$$

< identificateur 1 > est le nom de l'article

< identificateur 2 > désigne la partition qui structure l'article.

La zone qui suit permet d'indiquer la "place" de la partition ; deux cas sont possibles :

- CATAL

la partition figure dans un catalogue spécial ; le soin de l'insérer dans le programme est laissé au compilateur.

On évite ainsi à l'utilisateur d'avoir à décrire une même partition dans plusieurs programmes (2-4-6).

– Rien

il est nécessaire de décrire la partition dans le programme (phrases de type 3 ou 4).

< place > désigne le support de l'article :

Supports internes { MEMØIRE
STØCK
BANDE

Supports externes { ECARTE
SCARTE
ERUBAN
SRUBAN
CLAVIER
PAPIER
IMPRIME

Si cette indication est omise, il faut entendre que le support est MEMØIRE.

Dans le cas où la place désignée est un support externe, tous les objets, composant l'article doivent être de nature MØT.

2.3.4.2 - FILE

Le format général pour la description d'une file est :

{ DØNNEE
MIXTE
RESULTAT
AUXILIAIRE } & FILE & < identificateur 1 > & < longueur file > &
< identificateur 2 > & [CATAL] & } < nb > & [< identificateur-index >] {
[} & & & & & & & < nb > & [< identificateur-index >] } . . . [& [< place >]] ;

< identificateur 1 > est le nom de la file

< identificateur 2 > désigne la partition qui structure la file. La zone suivante est l'indication de la "place" de la partition (voir 2.4.6)

< longueur file > permet de donner deux indications :

- le nombre d'entrées de la file
- la longueur de chacune des entrées, i.e. le nombre d'éléments, de chacune des entrées

Il n'existe pas d'option par défaut pour ces longueurs.

La forme de < longueur file > est :

$$< \text{longueur 3} > \left[\left\{ \right\} < \text{longueur 3} > \left\{ \dots \right\} \right]$$

où < longueur 3 > peut être :

- une constance de nombre naturel,
- une variable de nature NATUREL et de statut DØNNEE ou MIXTE,
- un terme de longueur (2.7.10).
- rien, uniquement dans le cas où la variable est de statut DØNNEE, MIXTE ou RESULTAT. Le nombre d'éléments de l'entrée est alors fixé à l'exécution au moment de l'appel de procédure.

< nb > est un nombre naturel qui désigne le numéro de l'entrée (les entrées sont numérotées 1, 2, 3, ... etc); dans le cas où la file ne possède qu'une seule entrée, ce numéro peut être omis.

< identificateur-index > est défini par :

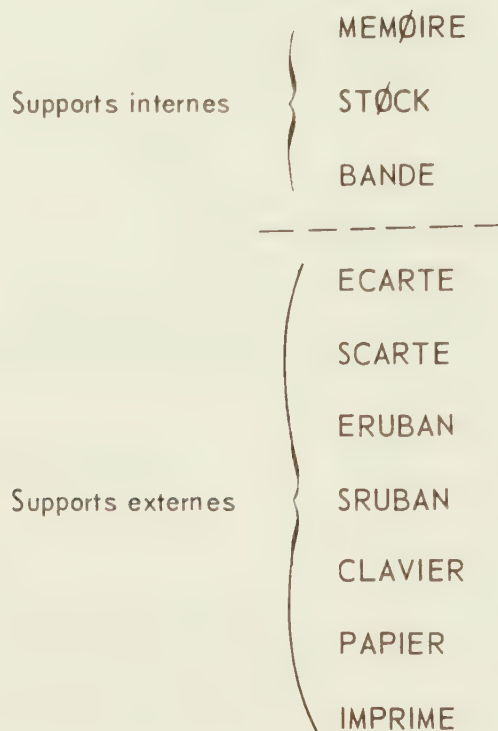
$$< \text{identificateur} > \left[\left\{ \right\} , < \text{identificateur} > \left\{ \dots \right\} \right]$$

il désigne les noms des index attachés à cette entrée de la file.

Il est possible de ne pas mentionner de nom d'index après le numéro de l'entrée ; dans ce cas il n'y a pas d'index attaché à cette entrée de la file et les éléments de cette entrée ne peuvent pas être individuellement désignés.

La zone composée des zones < nb > et < identificateur-index > est répétée autant de fois qu'il y a d'entrées, dans la file, auxquelles sont rattachés des index ; les entrées figurent dans ces zones dans l'ordre croissant des numéros.

< place > désigne le support de la file, MEMØIRE étant adoptée en cas d'omission.



Dans le cas où la place désignée est un support externe tous les éléments composant la file doivent être de nature MØT.

Le nombre d'entrées de la file n'est pas limité pour les files de support MEMØIRE.

Pour les files sur l'un des supports :

BANDE
STØCK
ECARTE
SCARTE
ERUBAN
SRUBAN
CLAVIER

le nombre d'entrées est limité à 1.

et pour les supports

PAPIER

IMPRIME

le nombre d'entrées peut être 1 ou 2 (la première est relative aux colonnes et la seconde - éventuelle - est relative aux lignes)

2.3.5 - Objets simples non-fondamentaux

2.3.5.1 - INDEX

Le format général pour la description d'un index est :

$\left\{ \begin{array}{l} D\emptyset N N E E \\ M I X T E \\ R E S U L T A T \\ A U X I L I A I R E \end{array} \right\}$	& INDEX & < identificateur > & < longueur >
	$\left[\& \left[\& \left[\& \left[\& \left[\text{MEMOIRE} \right] \right] \right] \right] \right] \right]$

< identificateur > est le nom de l'index.

Cet identificateur doit figurer dans la zone < identificateur-index > d'une description de file.

< longueur > est le nombre maximal de chiffres pouvant exprimer la position de l'index.

< longueur > peut être l'une quelconque des options définies en 2.3.3.1. pour les entiers naturels

Selon la valeur de l'expression de longueur, l'une des 2 représentations suivantes est adoptée pour l'index :

4 représentation sur 2 octets

9 représentation sur 4 octets

l'option par défaut est une représentation sur 4 octets.

Le statut d'un index est indépendant du statut et du support de la file à laquelle l'index est rattaché.

2.3.5.2 - PRØCEDURE

Le format général pour la description d'une variable de procédure est :

$$\left\{ \begin{array}{l} DØNNEE \\ AUXILIAIRE \end{array} \right\} \& \text{ PRØCEDURE } \& \langle \text{identificateur} \rangle$$

$$[\& [\& [\& [\& [\text{MEMØIRE}]]]]]] ;$$

$\langle \text{identificateur} \rangle$ est un nom désignant une procédure.

En d'autres termes, la valeur d'une variable de procédure est une procédure.

Les contraintes, imposées par le système d'exploitation, limitent à 8 caractères le nom d'une procédure, (sur les 30 caractères possibles seuls les 8 premiers sont retenus).

Toutes les procédures étant en mémoire au moment de l'exécution *, la place d'une procédure ne peut être que MEMØIRE; il doit en être de même pour la place de la variable qui la désigne.

Pour des raisons de simplification et de clarté, le statut des variables de procédure a été restreint à :

DØNNEE

AUXILIAIRE

Une variable de procédure de statut AUXILIAIRE doit obligatoirement faire l'objet d'une définition en phrase de type 5 (2.6).

Le nom d'une procédure (i.e. la valeur de la variable désignée par $\langle \text{identificateur} \rangle$) ne peut commencer par l'un des mots :

TRAITEM [ENTIMMEDIAT]

MACRØ

2.3.5.3 - ØRDRE

Le format général pour la description d'une variable de nature ORDRE est :

* La technique d' "overlay" ne contredit pas cette affirmation

$$\left(\begin{array}{l} \text{DONNÉE} \\ \text{MIXTE} \\ \text{RESULTAT} \\ \text{AUXILIAIRE} \end{array} \right) \& \emptyset \text{RDRE} \& \langle \text{identificateur} \rangle$$

$$[\& [\& [\& [\& [\text{MEMOIRE}]]]]]] ;$$

Ordre signifie ordre lexicographique des caractères (et non des symboles) de l'alphabet du langage (2.1.2).

Une seule variable de nature $\emptyset$ RDRE est autorisée dans un programme (i.e. traitement immédiat ou procédure).

Par définition, dans le cadre de l'application présente la longueur d'un objet de nature $\emptyset$ RDRE, i.e. le nombre de caractères, est 256.

Il est difficile de définir la valeur d'une variable de nature $\emptyset$ RDRE, qui est une relation entre caractères à l'aide d'une instruction d'affectation ou d'une instruction de changement de support du fait de la difficulté de perforation de l'ensemble des codes EBCDIC. Cette possibilité est toutefois conservée.

Ce type de variable est utilisé par les comparaisons entre objets de nature M $\emptyset$ T en utilisant un ordre lexicographique des caractères qui diffère de l'ordre standard (EBCDIC).

Les transcodages pour comparaison sur l'alphanumérique (lettres + chiffres) étant beaucoup plus classiques, un transcodage partiel a été défini.

L'alphabet du langage est divisé en trois classes :

– les lettres $\square$ ABCDEFGHIJKLMNOPQRSTUVWXYZ

– les chiffres 0123456789

– les symboles
restants . < (+ | ! \$ * ') ; - / , % > ? : # @ ' = "

soient :

C_L l'ensemble ordonné $\{ \square, A, B, C, \dots, Z \}$ ayant $N_L = 27$ éléments,

C_C l'ensemble ordonné $\{ 0, 1, 2, \dots, 9 \}$ ayant $N_C = 10$ éléments,

et C_R l'ensemble ordonné $\{ ., <, (, \dots, =, " \}$ ayant $N_R = 23$ éléments.

Ces ensembles sont dits ordonnés du fait que leurs éléments vérifient la relation :

$$e_j < e_k \quad \forall k > j \quad k, j \in \{1, N_i\} \quad \text{et } i = L, C, R$$

signifiant "est avant".

Les règles pour la définition d'un ordre sont :

1. le nombre de caractères de la définition ne peut être que :

- inférieur à 60 ;
- égal à 60 ;
- égal à 256.

2. Dans le cas où le nombre de caractères de la définition est inférieur à 60, les conditions suivantes s'appliquent :

1 - le nombre de caractères ne peut être que :

$$27 \\ \text{ou } 37$$

1.1 - dans le cas où le nombre est 27, l'ensemble des caractères de la définition ne peut être que :

$$\overline{C_L}$$

$\overline{C_L}$ (étant l'ensemble non ordonné des lettres, plus le blanc).

1.2 - dans le cas où le nombre est 37, l'ensemble des caractères de la définition ne peut être que :

$$C_{LC} = \{ \overline{C_L, C_C} \}$$

C_{LC} étant l'ensemble des lettres (plus le blanc) et des chiffres non ordonné.

3. Dans le cas où le nombre de caractères de la définition est 60, l'ensemble des caractères ne peut être que :

$$C_{LCR} = \{ \overline{C_L, C_C, C_R} \}$$

C_{LCR} étant l'ensemble des caractères de l'alphabet non ordonné.

4. Dans le cas où le nombre de caractères de la définition est 256, la condition suivante doit être satisfaite :

L'ensemble des caractères de la définition ne peut être que E, c'est-à-dire l'ensemble des caractères EBCDIC non ordonné.

Ces restrictions ont pour but d'éviter des ambiguïtés lors du transcodage.

2.3.5.4 - ETIQUETTE

Le format général pour la description d'une étiquette est :

AUXILIAIRE & ETIQUETTE & < identificateur > [& [& [& [& [MEMØIRE]]]]]] ;

Une étiquette est un élément interne à une procédure, en conséquence son statut ne peut être qu'AUXILIAIRE et sa place que MEMØIRE.

< identificateur > est le nom de l'étiquette.

Non seulement, il n'est pas nécessaire de décrire toutes les étiquettes qui apparaissent dans une procédure, on peut même n'en décrire aucune ce qui revient à dire que ce type de description est superflu. On l'a pourtant conservé en considérant des avantages que certains utilisateurs peuvent y trouver. Mais lorsqu'une ou plusieurs étiquettes figurent dans ces descriptions, elles doivent nécessairement apparaître dans la procédure.

2.3.5.5 - PARTITIØN

Le format général pour la description d'une partition, ou plus précisément d'une sous-partition, est :

$$\left\{ \begin{array}{l} DØNNEE \\ AUXILIAIRE \end{array} \right\} \& PARTITIØN \& \< \text{identificateur 1} \> \& \& \< \text{identificateur 2} \> \& [CATAL] ;$$

La partition désignée par < identificateur 1 > et dont la place est mentionnée dans la zone "place partition" est une sous-partition de la partition désignée par < identificateur 2 >

La signification de ce type de variable est expliquée au paragraphe 2.4.5.

2.3.6 - Règles générales concernant les descriptions

1. Tout objet traité dans un programme, hormis certaines partitions décrites ou définies antérieurement ou par la suite ainsi que les éléments qui la composent, doit être préalablement décrit ; en d'autres termes :

Le nom de tout objet doit figurer dans une des deux premières zones "Nom" d'une phrase de type 2.

En particulier un index doit figurer en deux endroits différents :

- dans la zone "identificateur-index", ce qui permet de préciser l'entrée et la file auxquelles il est rattaché ;
- dans la zone "Nom" d'une autre phrase de type 2 (ou 3 ou 4), ce qui permet de préciser son statut .

2. Un objet ne peut être décrit plus d'une fois, en d'autres termes :

Un même identificateur ne peut figurer plus d'une fois dans la première zone "Nom" d'une phrase de type 2.

3. A ces deux règles, il convient d'en ajouter une troisième sans laquelle l'interprétation des appels de procédure serait malaisée :

Les descriptions des données, des mixtes, des résultats et des auxiliaires doivent figurer dans cet ordre parmi les phrases de type 2 d'une même procédure.

2.4 - PARTITIONS

Les partitions sont décrites par des phrases de type 3 ou de type 4.

Elles décrivent la structure des objets groupés (i.e. articles et files).

2.4.1 - Concept de partition

Les partitions sont des constantes de structure : cela signifie que les structures qu'elles décrivent ne peuvent pas varier au cours de l'exécution du programme. On verra plus loin que cela n'entrave aucunement la faculté de décrire ce qu'on nomme communément des "formats variables".

En conséquence une partition n'est pas un agrégat d'informations, ou d'objets simples mais plutôt l'ensemble des adresses et des types de chacun des éléments qui représente le groupement de ces informations.

Dans la version actuelle du langage, il n'existe aucune "arithmétique" des partitions, c'est-à-dire qu'il est possible d'utiliser une ou plusieurs partitions dans la description d'une autre partition. Cette opération équivaut à un produit d'arborescences.

Une partition peut se schématiser par une arborescence contenant les informations hormis pour la racine : type de l'élément et adresse de l'élément.

L'utilisation d'une autre partition, elle-même schématisée par une arborescence, dans la description d'une partition, équivaut à substituer l'arborescence au sommet correspondant.

Une partition décrite dans une phrase de type 2 comme une sous-partition d'une autre partition est un objet de nature un peu particulière.

Cette partition regroupe certains éléments de la partition dont elle est une sous-partition, c'est pourquoi il est préférable de la considérer comme définie plutôt que comme décrite. En effet, cette partition est **définie** à partir d'éléments connus qui sont les éléments d'une autre partition.

Bien que cette opération soit en réalité une **définition**, c'est pour des raisons de pure commodité que sont employées les phrases de type 3 ou 4 plutôt que les phrases de type 5.

Le statut d'une sous-partition est cité dans une phrase de type 2 : il ne peut être que **DONNEE** ou **AUXILIAIRE**.

Le statut **DONNEE** est nécessaire puisqu'une partition et à fortiori une sous-partition sont assimilables à des constantes, dont, par définition, on ne peut qu'utiliser la valeur.

Le statut AUXILIAIRE est utilisé pour **définir** une sous-partition avant de l'utiliser parmi les arguments DØNNEE d'une procédure ou lors de certaines opérations de changement de support.

2.4.2 - Partitions indépendantes

Certaines partitions pouvant être "décrites en parallèle" (2.4.4), les autres partitions sont qualifiées d' "indépendantes" ; elles sont décrites par des phrases de type 3 dont le format général est le suivant :

< identificateur 1 > & [< condition >] & [< drapeau >] & [< nb fois >] & < nature > &
 < longueur > & [< identificateur >] [} & < condition >] & [< drapeau >] & [< nb fois >] &
 < nature > & < longueur > & [< identificateur >] { ...] ;

identificateur 1 > est le nom de la partition décrite.

La signification des différentes zones :

- zone condition
- zone drapeau
- zone nb fois
- zone nature
- zone longueur
- zone nom (< identificateur >)

est indiquée ci-après.

On distingue deux grandes classes d'éléments selon le contenu de la zone condition.

2.4.2.1 - ELEMENTS INCONDITIONNELS

La description d'un élément inconditionnel s'exprime dans l'un des formats suivants :

Format 1 - Elément simple

zone condition zone drapeau zone nb fois zone nature zone longueur zone nom
 .. & & & & < nature 1 > & < longueur > & [< identificateur >] ..

Format 2 - Elément répété 1ère forme

& & & < nb fois > & < nature 1 > & < longueur > & [< identificateur >]

Format 3 - Elément répété 2ème forme

& & < drapeau > & & < nature 2 > & < longueur > & [< identificateur >]

Format 4 - Elément doublement répété

& & < drapeau > & < nb fois > & < nature 2 > & < longueur > & [< identificateur >]

2.4.2.1.1 - ELEMENT SIMPLE - FORMAT 1

C'est un élément n'ayant qu'une occurrence dans le groupe d'éléments décrit.

2.4.2.1.2 - ELEMENT REPETE 1ère FORME - FORMAT 2

C'est un élément répété un certain nombre de fois indiqué par < nb fois >.

< nb fois > peut avoir les valeurs suivantes :

- une constante de nombre naturel ;
- une variable désignant un nombre naturel.

Dans le cas où < nb fois > est exprimé par une variable désignant un nombre naturel, celle-ci doit satisfaire les conditions énoncées au paragraphe 2.4.2.6.1.

En ce qui concerne les éléments simples et les éléments répétés 1ère forme, le format détaillé des zones nature 1, longueur et nom est :

zone Nature 1		zone longueur		zone nom
& MØT	&	< longueur 1 >	&	[< identificateur >]
& BØØLEEN	&	< longueur 1 >	&	[< identificateur >]
& NATUREL	&	< longueur 1 >	&	[< identificateur >]

zone Nature 1		zone longueur		zone nom	
&	ENTIER	&	< longueur 1 >	&	[< identificateur >]
&	BLANC	&	< longueur 1 >	&	[< identificateur >]
&	ZERØ	&	< longueur 1 >	&	[< identificateur >]
&	DECIMAL	&	< longueur 2 >	&	[< identificateur >]
&	REEL	&	< longueur 3 >	&	[< identificateur >]
&	ADLIB	&	< longueur 1 >	&	[< identificateur >]
&	RIEN	&		&	
&	PARTITION	&	[CATAL]	&	[< identificateur >]
&	INDEX	&	< longueur 4 >	&	[< identificateur >]
&	ETIQUETTE	&	< longueur 1 >	&	[< identificateur >]
&	PRØCEDURE	&	< longueur 1 >	&	[< identificateur >]

Dans le cas où < identificateur > qui est le nom désignant l'élément, est absent il ne sera pas possible de nommer l'élément dans la suite du programme.

< longueur 1 > est soit :

- un terme de longueur (2.7.10);
- une constante de nombre naturel ;
- une variable désignant un nombre naturel ;
- rien

Dans le cas où la longueur est fixée par une variable, celle-ci doit satisfaire les conditions exprimées au paragraphe 2.4.6.

Lorsque la longueur n'est pas fixée, deux cas se présentent :

- la partition structure un article ou une file de statut DØNNEE, MIXTE ou RESULTAT ; alors longueur est fixée à l'exécution au moment de l'appel de procédure . c est celle de l'élément homologue dans le programme appelant.
- la partition structure un article ou une file de statut AUXILIAIRE ; alors la longueur prend la valeur standard correspondant à la nature de l'objet.

Les conventions suivantes ont été adoptée, en fonction de la nature des objets :

1. ENTIER-NATUREL

Les conventions sont les mêmes que dans les phrases de type 2. Rappelons les :

Soit L l'expression de longueur.

1. Entier

$L \leq 4$ représentation sur 2 octets

$L \leq 9$ représentation sur 4 octets

2. Naturel

$L \leq 2$ représentation sur 1 octet

$L \leq 4$ représentation sur 2 octets

$L \leq 9$ représentation sur 4 octets

Toute valeur de L en dehors de ces limites, entraîne une erreur qui est détectée à la compilation ou à l'exécution.

2. BØØLEEN

La longueur d'un nombre booléen est arrondie au multiple de 8 immédiatement supérieur (ou égal), comme pour les phrases de type 2. La longueur maximale qui peut être spécifiée pour un booléen est 2048

3. MØT

La longueur maximale d'une variable de mot est 256.

4. BLANC

Il s'agit d'une suite d'occurrences du caractère (ou symbole) de l'alphabet ' ' (blanc)

Ainsi la description :

_____ & BLANC & 6 & _____

désigne un mot composé de 6 caractères ' ' .

La longueur maximale d'un tel objet est 256

5. ZERØ

Il s'agit d'une suite d'occurrences du caractère (ou symbole) de l'alphabet 0 ('F0').

Ainsi la description :

_____ & ZERØ & 9 & _____

désigne un mot composé de 9 caractères '0'.

La longueur maximale d'un tel objet est 256.

6. DECIMAL

La longueur d'un nombre décimal est composée de deux parties (2.3.3.3) dont la première est relative à la partie entière et la seconde à la partie décimale, la somme de ces deux longueurs ne peut être supérieure à 31. L'expression < longueur 2 > est définie par :

$$\text{longueur 1} > \quad . \quad < \text{longueur 1} >$$

et si les deux termes sont absents il est possible de supprimer le point décimal.

La longueur de la zone affectée à un nombre décimal est l'entier immédiatement supérieur à la demi-somme des deux expressions de longueur plus 1. Soit :

$$L_Z = \frac{L_1 + L_2 + 1}{2} \quad \text{ARRONDI A UNITE SUPER}$$

En conséquence la longueur maximale de la zone est 16 octets $((31 + 1)/2)$.

7. REEL

L'expression de longueur pour un nombre réel se compose de deux parties, l'une relative à l'ordre de grandeur de l'exposant et l'autre à la précision de la mantisse, comme on l'a dit à propos des phrases de type 2.

L'expression < longueur 3 > est définie par :

$$< \text{longueur 1} > \quad , \quad < \text{longueur 1} >$$

Lorsque les deux termes sont absents, il est possible de supprimer la virgule.

La longueur spécifiée pour l'exposant peut servir pour les changements de support (2.9).

La longueur spécifiée pour la mantisse peut aussi servir pour les changements de support, mais son rôle principal est de déterminer le format machine adopté pour la représentation du nombre :

long. ≤ 7 représentation simple précision (4 octets)

$8 \leq \text{long} \leq 17$ représentation double précision (8 octets)

8. ADLIB

Comme il a été mentionné (2.1.7.1.1), le mot réservé ADLIB désigne l'ensemble des objets simples fondamentaux, i.e. :

MØT

NATUREL

ENTIER

BOOLEEN

DECIMAL

REEL

ADLIB mentionné dans une phrase de type 3 désigne une partition dont les éléments sont des objets simples fondamentaux.

< longueur 1 > indique la longueur de cette zone (en octets)

Un élément de nature ADLIB peut être utilisé dans les instructions d'affectation et les termes de comparaison (2.7.9) . La longueur maximale qui peut être spécifiée pour un tel élément est limitée à 256.

Un élément de nature ADLIB est une suite d'informations sans structure particulière, sur lesquelles, donc, on ne peut effectuer aucune opération arithmétique

9. RIEN

RIEN indique l'absence d'élément.

RIEN est surtout utilisé dans la description de partitions en parallèle (phrases de type 4 - voir 2.4.4)

10. PARTITION

L'utilisation d'un élément de nature PARTITION permet d'inclure, dans la partition, la description de la partition désignée par < identificateur > ; cette partition peut être cataloguée (présence de CATAL dans la zone longueur). Cette opération équivaut à un produit d'arborescences.

La structure des éléments décrite par la partition citée est adjointe à la structure en cours de description.

Les descriptions récurrentes de partitions sont interdites ; on reviendra sur ce point au paragraphe 2.4.2.6.

11 - INDEX

La faculté d'utiliser un élément de nature INDEX permet de satisfaire d'une autre manière à la condition exprimée au paragraphe 2 3 6 : à savoir qu'une variable d'index doit avoir deux occurrences dans les descriptions :

- l'une dans la description de la file, qui permet de préciser deux éléments :
 - la file à laquelle il est rattaché ,
 - l'entrée de cette file à laquelle il est rattaché,
- l'autre soit en phrase de type 2, soit en phrase de type 3 ou 4.

Il est possible d'utiliser le facteur de répétition < nb fois > (2.4.2.1.2); on peut ainsi doter l'entrée de la file à laquelle l'index est rattaché (par son nom) de plusieurs index par l'utilisation des expressions qualifiées (2.5), e.g.

AUXILIAIRE & FILE & ALBERT & 20 & PARTITION 1 & CATAL & 1 & MAX & MEMOIRE ;

AUXILIAIRE & ARTICLE & MAXIMUM & & PARTINDEX & ;

PARTINDEX &&& 4 & INDEX & 4 & MAX ;

La file à une entrée désignée par ALBERT est dotée de 4 index, respectivement :

MAX (1) □ DE □ MAXIMUM

MAX (2) □ DE □ MAXIMUM

MAX (3) □ DE □ MAXIMUM

MAX (4) □ DE □ MAXIMUM

< longueur 4 > désigne le nombre maximal de chiffres décimaux nécessaire à exprimer la position de l'index et les conventions sont identiques à celles qu'on a énoncées pour les type 2, à savoir :

$long \leq 4$ représentation sur 2 octets

$5 \leq long \leq 9$ représentation sur 4 octets

Si la longueur est absente, dans le cas où la partition structure un article ou une file de statut AUXILIAIRE, une représentation sur 4 octets est adoptée.

12 - ETIQUETTE

L'utilisation d'un élément de nature ETIQUETTE permet de définir une variable de nature ETIQUETTE. Il est nécessaire que les articles ou les files structurés par une partition comprenant des éléments de nature ETIQUETTE soient tous de statut AUXILIAIRE et de support MEMOIRE pour respecter les principes énoncés au paragraphe 2.3.5.4.

Les éléments de nature ETIQUETTE doivent être définis par des phrases de type 5.

En employant le facteur de répétition < nb fois > (2.4.2.1.2) on peut définir une suite d'étiquettes, ce qui permet d'utiliser ultérieurement des branchements calculés : e.g.

AUXILIAIRE & ARTICLE & BRANCH && BRANCH1 && ;

AUXILIAIRE & NATUREL & 1 & 2 & ;

BRANCH1 & & & 5 & ETIQUETTE & & GØTØ ;

GØTØ & ETI1 , ETI2 , ETI3 , ETI4 , ETI5 ;

PRØC, (A, B; C; I) ;

VERS □ GØTØ (I) ;

ETI5 :

ETI2 :

ETI4 :

ETI3 :

ETI1 :

13 - PRØCEDURE

L'utilisation d'un élément de nature PRØCEDURE permet de définir une variable de nature procédure. Il est nécessaire que les articles ou les files structurés par une partition comprenant de tels éléments soient de statut AUXILIAIRE ou DØNNEE et de place MEMØIRE pour respecter les principes énoncés au paragraphe 2.3.5.2.

Les éléments de nature PRØCEDURE doivent être définis par des phrases de type 5 (2.6) lorsqu'ils sont de statut AUXILIAIRE.

En employant le facteur de répétition < nb fois > on peut, comme pour les étiquettes définir une suite de procédures (plus exactement : une suite de noms de procédures), ce qui permet d'utiliser ultérieurement des appels de procédures calculés, e.g. :

AUXILIAIRE & ARTICLE & CALL & CALPART & & ;

AUXILIAIRE & NATUREL & X & 2 & ;

CALPART & & & 3 & PRØCEDURE & & APPEL ;

APPEL □ DE □ CALL & P1 , P2 , P3 ;

BØUCLE : X : = ;

APPEL (X) □ DE □ CALL, (— ; — ; —) ;

VERS □ BØUCLE ;

On peut spécifier une longueur pour les éléments de nature ETIQUETTE et PROCEDURE ; cette longueur représente le nombre maximal de caractères qui composent le nom de l'étiquette ou celui de la procédure désigné par la variable.

La valeur maximale de l'expression désignée par `< longueur 1 >` est limitée à 30 (nombre de caractères).

Il est rappelé que dans l'application considérée ici, seuls les huit premiers caractères du nom d'une procédure sont significatifs.

L'usage d'expressions de longueur n'a pas été introduit dans les descriptions d'étiquettes et de procédures en phrase de type 2 du fait que le contexte permet toujours de déterminer la place à réserver sans préoccupation du cadrage de l'élément. La situation est différente en ce qui concerne les partitions.

La longueur attribuée par défaut est 30.

2.4.2.1.3 - ELEMENT REPETE 2ème FORME - FORMAT 3

Cette forme de répétition est réservée exclusivement aux éléments de nature MØT ; d'une façon plus précise, le format 3 est le suivant :

& &< drapeau > & & MØT & < longueur 1 > & [`< identificateur >`]

Cela permet de décrire une suite d'éléments, de longueur désignée par `< longueur 1 >` (même définition qu'en 4.2.1.2), de nature MØT désignée par `< identificateur >` répétée jusqu'à l'occurrence d'un élément identique à l'élément de nature MØT cité dans `< drapeau >`.

`< drapeau >` est défini par :

`< drapeau 1 >` , `< longueur maxi >`

`< longueur maxi >`, qui ne figurait pas dans la première version du langage, est soit une constante de nombre naturel ou une variable désignant un nombre naturel ou encore rien si les articles et/ou les files que structure la partition sont de statut DØNNEE, MIXTE ou RESULTAT, dans ce cas la longueur maximale de la répétition est fixée à l'exécution au moment de l'appel de procédure. Si `< longueur maxi >` est absent `< drapeau >` devient :

`< drapeau 1 >` [,]

`< longueur maxi >` est le nombre maximal de répétitions de l'élément; il permet de réaliser l'allocation mémoire au moment de l'exécution.

drapeau 1 > est une constante de mot ou une variable de nature MØT et de statut soit AUXILIAIRE - et il doit alors faire l'objet d'une définition en phrase de type 5 (2.6) - soit DØNNEE (2.3.3.2).

La longueur du drapeau (i.e. nombre de caractères de < drapeau 1 >) et la longueur de l'élément (valeur spécifiée par < longueur 1 >) doivent être identiques.

Dans le cas où < longueur 1 > n'est pas spécifié, le statut des articles et/ ou des files structurés par la partition ne peut être que DØNNEE, MIXTE ou RESULTAT ; la longueur de l'élément est fixée et comparée à celle du drapeau, à l'exécution, au moment de l'appel de procédure. e.g.

— & ' * " , 25 & & MØT & 2 & TEXTE

désigne une suite - peut être vide - de deux caractères (mot de longueur 2) TEXTE (1), TEXTE (2), etc... qui s'arrête à la rencontre du premier groupement de caractères ' / * ". La longueur maximale de cette répétition est 25 (i.e. de TEXTE (1) à TEXTE (25). Le mot ' / * " n'est pas compris dans la suite.

2.4.2.1.4 - ELEMENT DOUBLEMENT REPETE - FORMAT 4

Ce type de répétition est la combinaison des deux précédents, il est réservé aux éléments de nature MØT ; le format 4 est donc le suivant :

& & < drapeau > & < nb fois > & MØT & < longueur > & [< identificateur >]

Cette forme décrit une suite de groupes de < nb fois > éléments de nature MØT, ces groupes étant répétés jusqu'à la rencontre d'un élément identique à l'élément de nature MØT désigné par < drapeau 1 >.

L'ensemble des conditions pour la description d'un tel type est l'ensemble des conditions formulées pour les éléments de 1ère et de 2ème forme.

Ainsi

— & & ' / * / " , 12 & 5 & MØT & 3 & SERIE —

désigne une succession de groupes de 3 caractères, répétition qui prend fin dès qu'apparaît le groupement ' / * / " ; il y a au plus 60 groupes (5.12).

Ainsi qu'il est précisé par la suite (2.5.1.4), l'ensemble de ces mots est désigné par SERIE ; la première et la seconde suite (si elles existent) sont désignées par SERIE (1) et SERIE (2) ; et les mots eux-mêmes (i.e. 3 caractères), dans l'ordre par :

SERIE (1) (1), SERIE (2) (1), SERIE (3) (1), SERIE (4) (1), SERIE (5) (1), SERIE (1) (2), SERIE (2) (2) , SERIE (3) (2), etc...

2.4.2.2 - PARTITIONS SEQUENTIELLES

Une partition séquentielle est une suite d'éléments inconditionnels. On la décrit par une phrase de type 3 composée successivement :

- du nom de la partition en zone nom de partition (< identificateur 1 >) ;
- de la description du 1er élément dans les zones suivantes ;
- des descriptions des éléments suivants, l'un après l'autre, dans les zones correspondantes ;
- d'un point virgule ' ; ' " après la dernière zone < nom >, qui met fin à l'énumération.

Ainsi :

Zone Nom de partition	Zone Condition	Zone Drapeau	Zone nb fois	Zone Nature	Zone longueur	Zone Nom
EXEMPLE &	&	&	25	& RÉEL	& 2,7	& FLØAT &
&	&'ABC+ ',LG&	&	6	& MØT	& 4	& CØM &
&	&	&	5	& PRØCEDURE	& 10	& PRØC &
&	& ' ; ' , 30	&		& MØT	& 1	& SUITE &
&	&	&		& NATUREL	& 9	& BLØC ;

Les conditions que doit remplir la description d'une partition séquentielle sont exposées au paragraphe (2.4.2.6).

2.4.2.3 - ELEMENTS CONDITIONNELS

2.4.2.3.1 - DEFINITION

Un élément conditionnel d'une partition est un élément qui n'apparaît que si la condition formulée dans la zone condition est remplie.

Le format général pour la description d'un élément conditionnel est :

— & < condition > & [< drapeau >] & [< nb fois >] & < nature > & < longueur > &
[< identificateur >] —

Contrairement à la première version du langage, on a restreint ici, la portée d'une condition à l'élément qui suit immédiatement la condition, ainsi :

```
___ & K = 1 & & 5 & DECIMAL & 16.2 & BENEFICE & & & & DECIMAL &
2.3 & TAUX & ___
```

L'élément BENEFICE n'est présent que si $K = 1$, alors que l'élément TAUX est présent, dans les articles et/ou les files qui structure la partition, quelle que soit la valeur de K

Cette restriction a été formulée par souci de clarification de la description d'une partition.

2.4.2.3.2 - TYPE DES CONDITIONS

Les conditions qui figurent dans la zone condition peuvent être de l'un des types suivants :

2.4.2.3.2.1 - Condition du type arithmétique

On l'écrit ainsi :

- une variable, suivie de :
- l'un des symboles de comparaison :

$=$, $<$, $>$, $\neq$, $>=$ ou $\neq$

suivi d'

- une variable ou d'une constante.

Les deux variables ou la variable et la constante doivent désigner des objets de même nature. La nature des objets qu'il est possible de faire figurer dans les conditions ne peut être que :

MØT

BØØLEEN

NATUREL

ENTIER

DECIMAL

Si le même élément conditionnel correspond aux valeurs b , c et d , par exemple, de la variable a , il est possible d'abrégé l'écriture. Ainsi, au lieu de :

____ & a < symbole de comparaison > b & < élément >
 & a < le même symbole > c & < le même élément >
 & a < le même symbole > d & < le même élément > ____

On peut écrire :

____ & a < symbole de comparaison > b | c | d & < élément > ____

2.4.2.3.2.2 - Condition du type booléen

– c'est une variable qui désigne un nombre booléen.

La condition est satisfaite si tous les éléments du nombre booléen sont vrais (VRAI ou 1).

Le type de condition :

< nombre naturel > IEME

qui figurait dans la première version du langage a été supprimé.

Cette forme exprimait une suite de conditions du type :

& K = 1 & ____
 . . .
 & K = 2 & ____
 . . .
 & K = 3 & ____

sans mentionner la variable K.

Il a semblé plus judicieux de généraliser les types de conditions, tout en mentionnant explicitement les variables de condition.

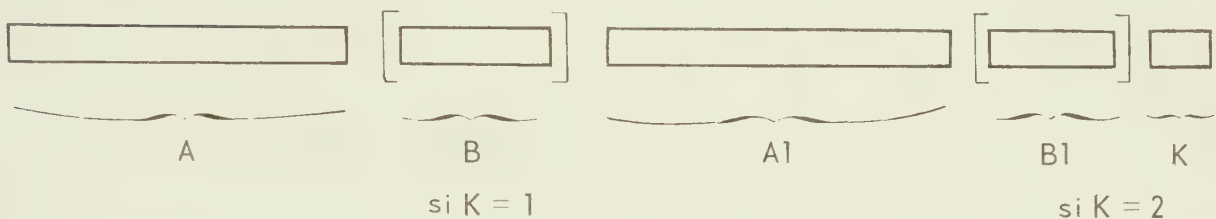
Les objets de nature REEL ne sont pas autorisés dans l'expression des conditions du fait de l'imprécision concernant le test d'égalité en virgule flottante, cette imprécision étant due à la représentation des nombres.

2.4.2.3.3 - REGLES CONCERNANT LES DESCRIPTIONS D'ELEMENTS CONDITIONNELS

Pour qu'il soit possible d'opérer sur les éléments conditionnels, il faut connaître, sans ambiguïté, la position de la (ou des) variable (s) de condition. En effet, soit la description :

PART	&		&	&	&	MØT	&	10	&	A	&
	&	K	1	&	&	REEL	&	2,7	&	B	&
	&		&	&	&	MØT	&	20	&	A1	&
	&	K - 2	&	&	&	REEL	&	2,7	&	B1	&
	&		&	&	&	NATUREL	&	4	&	K	;

Le schéma en mémoire d'un article structuré par cette partition, est le suivant :



Pour connaître la structure en mémoire de l'article, il faut nécessairement connaître la valeur de K, donc son emplacement dans l'article. Or cet emplacement varie avec la valeur de K ; cela rend le traitement long et complexe, voire aléatoire.

Il faut donc imposer certaines contraintes, qui peuvent d'ailleurs rendre plus claire, pour l'utilisateur, la description des partitions.

Considérons tout d'abord les conditions exprimées de l'une ou l'autre des manières suivantes :

- variable (booléenne)
- variable < symbole de comparaison > nombre

Deux cas sont à distinguer :

1er Cas :

La variable de condition est décrite dans la partition, il faut alors que :

- la déclaration précède l'utilisation
- la déclaration soit inconditionnelle.

2ème Cas :

La variable de condition n'appartient pas à la partition, il faut alors que :

- la déclaration de la variable précède tout emploi, dans une autre partition, où elle est utilisée ;

En conséquence l'emploi isolé de la partition pour structurer un article ou une file est interdit.

- dans le cas où la déclaration de la variable est conditionnelle, les partitions qui utilisent cette variable, ne doivent apparaître que si la variable est apparue.
En conséquence ces partitions et la variable de condition doivent être soumises à des conditions portant sur la même variable.

Considérons ensuite les conditions exprimées sous la forme :

variable < symbole de comparaison > variable

Deux cas sont à distinguer :

1er Cas :

Les variables de condition appartiennent à la partition. Il faut alors que :

- les déclarations précèdent l'utilisation ;
- les déclarations soient inconditionnelles.

2ème Cas :

L'une des deux variables (ou les deux variables) n'appartient pas à la partition. Il faut alors que :

- les déclarations des variables de condition précèdent tout emploi, dans une autre partition, elles sont utilisées.
Cette même condition se formule pour le cas d'une seule variable non déclarée dans la partition. En conséquence l'emploi isolé de la partition pour structurer un article ou une file est interdit.
- dans le cas où les déclarations des variables de condition sont conditionnelles, les partitions qui utilisent ces variables ne doivent apparaître que si ces dernières sont apparues.
En conséquence ces partitions et les variables de condition sont soumises à des conditions portant sur la même variable.
Cette même condition se formule pour le cas d'une seule variable non déclarée dans la partition.

2.4.2.4 - ELEMENTS COMBINES

Il est possible de mentionner des facteurs de répétition pour les éléments conditionnels.

Les formats généraux sont :

zone condition	zone drapeau	zone nb fois	zone nature	zone longueur	zone nom
< condition >	&	&	& < nature 1 >	& < longueur >	& [< identificateur >]
< condition >	&	& < nb fois >	& < nature 1 >	& < longueur >	& [< identificateur >]
< condition >	& < drapeau >	&	& MØT	& < longueur 1 >	& [< identificateur >]
< condition >	& < drapeau >	& < nb fois >	& MØT	& < longueur 1 >	& [< identificateur >]

Les conditions auxquelles doivent satisfaire les différents éléments figurant dans les zones "Condition", "Drapeau", "Nb fois" sont les conditions énoncées jusqu'ici et celle qu'on exposera au paragraphe 2.4.2.6.

2.4.2.5 - PARTITIONS CONDITIONNELLES

Une partition conditionnelle est une suite d'éléments conditionnels et inconditionnels.

Elle est décrite par une phrase de type 3 en plaçant :

- le nom de la partition en zone "nom de partition" (< identificateur 1 >) ;
- la description du 1er élément dans les zones suivantes ;
- les descriptions des éléments suivants, l'un après l'autre, dans les zones correspondantes ;
- un point virgule. ' ; ' , après la dernière zone "nom", qui met fin à l'énumération

e.g.

EXEMPLE 2 & & & 15 & NATUREL & 2 & ALPHA &
 & & & & ENTIER & 3 & K1 &
 & & & & REEL & 2,7 & BETA &
 & K1 = 0 & & & ENTIER & 3 & K2 &
 & K1 = 2 & & & PARTITION & & EXEMPLE 3 &
 & & & 10 & NATUREL & 2 & GAMMA ;

EXEMPLE 3 & & & & NATUREL & 2 & DELTA &
 & K2 = 1 & & & MØT & 256 & CØMMENT &
 & K2 > K1 & & & ENTIER & 4 & EPSIL1 &
 & K1 = K2 & & & ENTIER & 4 & EPSIL2 ;

2.4.2.6 - CONDITIONS GENERALES CONCERNANT LA DESCRIPTION DES PARTITIONS

Il est interdit de décrire les partitions d'une façon récurrente.

En d'autres termes, une description de partition ne doit pas contenir de référence à elle-même que ce soit dans sa propre description ou dans l'une quelconque des partitions qu'elle utilise pour sa description ; Ainsi la suite des descriptions ci-dessous est interdite.

PART1 & & & & MØT & 10 & A1 &
 & & & & NATUREL & 4 & B1 &

& & & & PARTITION & & PART2 &

& & & & REEL & 2,17 & X1

PART2 & & & & ENTIER & 9 & I1 &
 & & & & PARTITION & & PART3 &

PART3 & & & & REEL & 2,17 & X2 &
 & & & & PARTITION & & PART1 &

En effet, la description de PART1 est récurrente, puisque, l'un des éléments des partitions qu'elle utilise est cette partition même. Une telle écriture est dépourvue de signification.

Plusieurs cas se présentent :

1. < Nb fois > est un identificateur :

1.1. Cet identificateur figure dans les phrases de type 2 d'un programme. Il est alors nécessaire que l'objet désigné par cet identificateur soit de statut ~~D~~ONNEE, ou de statut AUXILIAIRE faisant l'objet d'une définition en phrase de type 5 (2.6). En effet, l'article ou la file que structure la partition est engendré en mémoire lors de l'exécution ; la longueur de la répétition, i.e. la valeur de l'objet désigné par l'identificateur, doit donc être connue à ce moment.

1.2. Cet identificateur figure dans les phrases de type 3 ou 4 du programme. Ce cas se ramène au cas suivant :

2. < Nb fois > est une expression qualifiée

Comme on l'a indiqué au paragraphe 2.5, les objets constituant un article ou une file, objets qui sont donc structurés par une partition, ont le même statut et le même support que l'article ou la file dont ils sont extraits.

Le principe à observer est le suivant :

Il faut que la valeur de l'objet désigné par l'expression qualifiée soit connue lors de l'exécution, soit au moment de l'appel de procédure soit à celui du lancement d'un travail par un "traitement immédiat".

En conséquence, selon les cas, l'une ou l'autre des règles suivantes doit être observée :

- Si l'objet désigné par l'expression qualifiée est de statut argument, ce statut est obligatoirement ~~D~~ONNEE et le support ne peut être que ~~M~~EMOIRE.
- Si l'objet désigné par l'expression qualifiée est de statut AUXILIAIRE, il doit faire dans ce cas l'objet d'une définition en phrase de type 5 (2.6) ; ceci entraîne que son support doit être ~~M~~EMOIRE.

Dans le cas d'un traitement immédiat, les seules façons d'exprimer < Nb fois > sont les suivantes :

- une constante de nombre naturel ;
- une variable (identificateur ou expression qualifiée), variable qui désigne un objets (de statut AUXILIAIRE par définition (2.11) de nature NATUREL et dont le support est ~~M~~EMOIRE ; de plus, elle doit obligatoirement faire l'objet d'une définition en phrase de type 5 (2.6)..

2.4.2.6.2 - ELEMENTS REPETES 2ème FORME

Rappel du format :

zone condition zone drapeau zone Nb fois zone nature zone longueur zone nom

& < drapeau > & & MØT & < longueur l > & [< identificateur >]

< drapeau > à la forme suivante :

< drapeau l > , < longueur maxi >

< drapeau l > étant :

- soit une constante de MØT
- soit une variable désignant un objet de nature MØT

et < longueur maxi > étant :

- soit une constante de nombre naturel
- soit une variable désignant un objet de nature NATUREL.

< drapeau l > doit satisfaire aux conditions suivantes :

1. Les conditions énoncées au paragraphe (2.4.2.1.3) concernant les descriptions des objets répétés 2ème forme (condition de longueur notamment).
2. Deux cas se présentent :
 - < drapeau l > est une constante de MØT. Aucune condition spéciale.
 - < drapeau l > est une variable désignant un objet de nature MØT.

Il y a alors plusieurs possibilités :

2.1. < drapeau l > est un identificateur :

2.1.1. Cet identificateur figure dans les phrases de type 2 du programme. Il est alors nécessaire que l'objet désigné par cet identificateur soit l'un ou l'autre des deux types suivants :

- objet de nature MØT, de statut AUXILIAIRE et de support MEMØIRE ;
- objet de nature MØT, de statut AUXILIAIRE, de support MEMØIRE et faisant l'objet d'une définition en phrase de type 5 (2.6).

2.1.2. Cet identificateur figure dans les phrases de type 3 ou 4 du programme. Ce cas se ramène au cas suivant.

2.2. < drapeau l > est une expression qualifiée :

Cette expression qualifiée doit alors satisfaire l'une ou l'autre des conditions suivantes selon le cas :

- Si l'objet désigné par l'expression qualifiée est un argument, alors son statut ne peut être DØNNEE et le support ne peut être que MEMØIRE.
- Si l'objet par l'expression qualifiée est de statut AUXILIAIRE, alors sa place ne peut être que MEMØIRE et il doit faire l'objet d'une définition en phrase de type 5 (2.6)

Dans le cas où le programme est un traitement immédiat, < drapeau 1 > ne peut être que :

- ou bien une constante de MØT ;
- ou bien une variable (de statut AUXILIAIRE par définition (2.11) de support MEMØIRE faisant obligatoirement l'objet d'une définition en phrase de type 5.

< longueur maxi > doit satisfaire aux conditions suivantes :

Deux cas se présentent selon que < longueur maxi > est :

1. Une constante de nombre naturel. Dans ce cas aucune condition spéciale n'est requise.
2. Une variable désignant un objet de nature NATUREL.

Cette variable peut être :

2.1. un identificateur :

- Cet identificateur apparait dans les phrases de type 2 du programme comme argument ; son statut ne peut donc être que DØNNEE et son support que MEMØIRE.
- L'objet désigné est de statut AUXILIAIRE, son support doit être MEMØIRE il doit faire l'objet d'une définition en phrase de type 5 (2.6).

2.2. une expression qualifiée :

- Si l'objet désigné est un argument, alors son statut ne peut être que DØNNEE et son support ne peut être que MEMØIRE.
- Si l'objet désigné est de statut AUXILIAIRE, son support doit être MEMØIRE et il doit faire l'objet d'une définition en phrase de type 5 (2.6).

Dans le cas où le programme est un traitement immédiat, < longueur maxi > ne peut être que :

- ou bien une constante de nombre naturel ;
- ou bien une variable (de statut AUXILIAIRE obligatoirement (2.11), de support MEMØIRE qui fait l'objet d'une définition en phrase de type 5.

Dans les deux cas précédents de répétition, (i.e. 1ère et 2ème forme) il faut observer une règle générale concernant la déclaration conditionnelle des éléments de l'une ou l'autre des formes suivantes :

Eléments répétés 1ère forme - < Nb fois >

Eléments répétés 2ème forme - < drapeau 1 > et/ou < longueur maxi >

Cette règle s'énonce ainsi :

Dans le cas où l'occurrence de ces facteurs de répétition est conditionnelle, les éléments répétés dans lesquels ils sont utilisés ne doivent apparaître dans le texte qu'après ces facteurs de répétition.

2.4.2.6.3 - ELEMENTS DOUBLEMENT REPETES

Rappel du format

Zone condition zone drapeau zone Nb fois zone nature zone longueur zone nom

& < drapeau > & < Nb fois > & MÔT & < longueur > & [< identificateur >]

Les conditions que doivent satisfaire < drapeau > et < nb fois > sont les conditions énoncées pour les éléments répétés 1ère forme (2.4.2.1.2) (< nb fois >) et les éléments répétés 2ème forme (2.4.2.1.3) (< drapeau >).

Dans le cas où l'occurrence des variables qu'il est possible de faire figurer dans les zone "Drapeau" et "Nb fois" est conditionnelle, il importe que les éléments où sont utilisées ces variables n'apparaissent dans le texte qu'après les variables.

2.4.2.6.4 - ELEMENTS COMBINES

Ce type d'élément doit satisfaire à toutes les conditions formulées pour les éléments qui servent à définir sa structure, i.e., suivant le cas :

- < condition > + < Nb fois >
- < condition > + < drapeau >
- < condition > + < drapeau > + < Nb fois >

2.4.3 - Cadrage des articles et des files placés en mémoire

Un article, ou une file, est toujours structuré par une partition. Lorsqu'un tel objet groupé se trouve en mémoire pour des opérations de calcul, on gagne sur le temps de traitement lorsque certains éléments sont cadrés (ou alignés).

Considérons par exemple le cas suivant :

Soit la partition PART 1 structurant l'article ART 1 :

```

PART 1  & & & & MØT      & 12  & GENERIQUE &
        & & & & REEL      & 2,17 & DØUBLE  &
        & & & & BØØLEEN & 24  & CLE      &
        & & & & ENTIER    & 6    & INDICE    &
        & & & & MØT      & 15    & INDIQFIN ; &
  
```

L'article est donc représenté en machine par la suite d'éléments ci-dessous :

12 caractères	GENERIQUE
1 double mot *	DØUBLE
3 caractères	CLE (ici 1 caractère = 8 bits)
1 mot	INDICE
15 caractères	INDIQFIN

Si l'article est représenté tel quel en mémoire, les instructions machine relatives aux opérations portant sur DØUBLE et INDICE ne peuvent être les instructions usuelles pour le traitement des nombres en virgule flottante double précision ou des nombres binaires (sur 1 mot) respectivement..

En effet, il faudrait que DØUBLE et INDICE fussent situés dans des zones intermédiaires de mémoire cadrées au double-mot et au mot respectivement.

Pour pallier cet inconvénient, un cadrage implicite doit être effectué lors de l'exécution, au moment de la génération de l'article (ou de la file) dans la zone des données (i.e. zone des objets traités) ; les éléments du type :

ENTIER

NATUREL

REEL

sont cadrés à des limites, qui sont respectivement :

- une limite de demi-mot pour les nombres naturels ou entiers représentés sur 2 octets.

* un mot = 4 caractères.

- une limite de mot pour les nombres naturels ou entiers représentés sur 4 octets et pour les nombres réels simple précision
- une limite de double-mot pour les nombres réels double précision

Ce cadrage implicite augmente la place occupée par les objets groupés en raison des zones de cadrage, qu'il intercale et qui ne sont pas utilisées par le traitement

Toutefois, l'utilisateur a la possibilité de définir un facteur de cadrage en tête des descriptions de partition, et ce grâce à l'élément de ces descriptions dont on n'a pas parlé jusqu'ici, à savoir `SYNCHRØ` ; le format général est :

	zone	zone	zone	zone	zone	zone
zone Nom de partition	condition	drapeau	nb fois	nature	longueur	nom
< identificateur >	&	&	&	& <code>SYNCHRØ</code> &	< facteur > &	_____

où < facteur > est :

- soit une constante de nombre naturel ;
- soit une variable désignant un nombre naturel.

Les conditions auxquelles doit satisfaire < facteur > dans le cas où il est exprimé par une variable sont identiques à celles que doit remplir < nb fois > dans le même cas ; de plus < facteur > ne peut désigner que l'une des valeurs suivantes :

1, 2, 4 ou 8

Toute valeur différente entraîne une erreur, laquelle est détectée à la compilation ou à l'exécution suivant la façon dont < facteur > est exprimé

La valeur 1 n'entraîne aucune modification du cadrage: l'article (ou les articles de la file) occupent la première place libre en mémoire quel qu'en soit le cadrage. En l'absence d'un élément `SYNCHRØ` en tête de la description d'une partition, la situation est identique à celle qui vient d'être décrite. En d'autres termes - `SYNCHRØ & 1 &` - est l'option par défaut.

La valeur 2 entraîne le cadrage de l'objet structuré à une adresse de demi-mot.

De même les valeurs 4 et 8 entraînent le cadrage de l'objet structuré à des adresses de mot et de double-mot respectivement.

Il est permis d'utiliser un élément de nature `SYNCHRØ` en tête de toute description de partition, que celle-ci structure directement ou non un article ou une file.

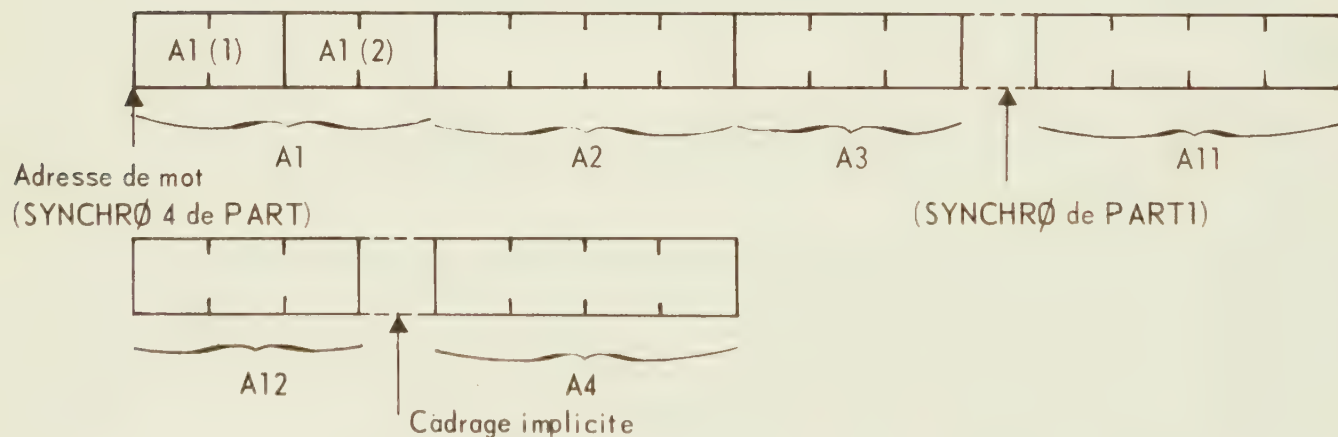
Dans le cas où la partition ayant en tête de sa description un élément SYNCHRO est utilisée dans la description d'une autre partition, l'objet structuré par cette dernière sera cadré pour les éléments de la partition utilisée selon la spécification donnée par SYNCHRO. e.g.

AUXILIAIRE & ARTICLE & A & & PART & ;

PART & & & & SYNCHRO & 4 & &
 & & & 2 & ENTIER & 4 & A1 &
 & & & & NATUREL & 6 & A2 &
 & & & & MOT & 3 & A3 &
 & & & & PARTITION & & PART1 &
 & & & & REEL & & 2,7 & A4 ;

PART1 & & & & SYNCHRO & 4 & &
 & & & 2 & REEL & & 2,6 & A11 &
 & & & & MOT & & 3 & A12 ;

Le schéma de l'article représenté en mémoire est :



La possibilité qu'on a de définir des objets de nature NATUREL représentés sur un octet, jointe à l'usage des éléments SYNCHRO, permet de décrire la partition de façon à éviter au maximum les cadrages implicites. grâce à ces dispositions, l'usager pourra, dans bien des cas, réduire de façon sensible la place occupée par les objets groupés.

Les articles et files placés sur les supports externes ont des éléments de nature MOT, par définition ; le problème de l'alignement ne se pose pas pour l'introduction en mémoire de ces objets (la phrase de type 4 décrivant éventuellement les transformations, que doit subir

l'article ou la file lors des opérations d'échange, peut provoquer les conversions et "l'éclatement" de l'article ou de la file à la lecture en mémoire, et les conversions et le regroupement à la sortie).

Pour les articles et les files placés sur les supports du type ~~ST~~OCK ou BANDE, deux représentations sont possibles :

- l'image mémoire ;
- la compression de l'objet

L'utilisateur peut choisir l'une ou l'autre :

- le stockage du type "image mémoire" occupe une place importante sur les supports externes, mais ne nécessite aucun traitement particulier lors des opérations d'échange.
- le stockage avec compression de l'objet, qui consiste à supprimer toutes les zones non significatives, occupe une place minimale sur les supports ~~ST~~OCK ou BANDE; mais il exige un surcroît de temps de traitement pour que soient respecté les contraintes d'alignement.

L'utilisateur choisira donc le mode de représentation en fonction de l'application particulière qu'il se propose.

Le compilateur, si l'usager le désire (3.1), écrira sur le fichier de sortie des "résultats" de compilation (en général sur l'imprimante) la carte des objets structurés (MAP), en bref leur géographie.

2.4.4 - Partitions décrites en parallèle

On peut décrire en parallèle deux partitions dont les éléments sont mis deux à deux en correspondance.

Ces descriptions en parallèle sont utilisées principalement pour :

- les opérations de conversion qui doivent être effectuées sur des objets groupés (articles ou files) placés en mémoire,
- les opérations de changement de support (2.9), d'une façon générale.

Les objets placés sur des supports externes sont obligatoirement de nature ~~M~~OT; une opération de lecture de cartes perforées peut avoir pour objet d'amener en mémoire des objets dont certains ne sont pas de nature ~~M~~OT mais NATUREL, ENTIER, etc... et dont les mots contenus dans la carte sont les noms. Cette opération est

en fait un ensemble de processus de conversion, il a semblé judicieux de résumer l'ensemble de ces opérations en une description en parallèle des structures des objets concernés par ces opérations.

D'une façon générale, la description en parallèle de partitions est une façon synthétique d'exprimer un ensemble d'opérations de conversion.

Les éléments de nature RIEN sont utilisés principalement dans les descriptions en parallèle de partitions où ils servent à supprimer des zones non significatives.

Les éléments mis en parallèle doivent satisfaire à certaines conditions concernant leurs natures dans le cas où celles-ci sont différentes.

Les conversions s'opèrent dans un sens déterminé, on est conduit à considérer le tableau suivant :

ENTRÉE S Ø R T I E	SYNCHRO Ø	MØT	BØØLEEN	NATUREL	ENTIER	BLANC	ZERØ	DECIMAL	REEL	ADLIB	RIEN	PARTITION ØN	INDEX	PROCEDURE
SYNCHROØ	Ø	N	N	N	N	N	N	N	N	N	Ø	N	N	N
MØT	N	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	N	Ø	Ø	Ø
BØØLEEN	N	Ø	Ø	Ø	N	N	N	N	N	Ø	N	Ø	N	N
NATUREL	N	Ø	Ø	Ø	Ø	N	N	Ø	Ø	Ø	N	Ø	N	N
ENTIER	N	Ø	Ø	Ø	Ø	N	N	Ø	Ø	Ø	N	Ø	N	N
BLANC	N	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	N	Ø	N	N
ZERØ	N	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	N	Ø	N	N
DECIMAL	N	Ø	N	Ø	Ø	N	N	Ø	Ø	Ø	N	Ø	N	N
REEL	N	Ø	N	Ø	Ø	N	N	Ø	Ø	Ø	N	Ø	N	N
ADLIB	N	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	N	Ø	N	N
RIEN	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	N
PARTITIONØN	N	Ø	Ø	N	N	Ø	Ø	N	N	Ø	N	Ø	N	N
INDEX	N	Ø	N	N	N	N	N	N	N	N	N	Ø	Ø	N
PROCEDURE	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Ø signifie mise en parallèle possible
N signifie mise en parallèle interdite

Les éléments de nature ETIQUETTE ne peuvent pas figurer dans la description de partitions en parallèle du fait qu'ils sont internes à une procédure.

Les éléments de nature PRØCEDURE peuvent figurer dans une telle description, la seule conversion permise étant :

élément de nature PRØCEDURE $\longrightarrow$ élément de nature MØT

Ces éléments pouvant appartenir à un article ou à une file de statut DØNNEE, il est nécessaire de pouvoir connaître leur valeur (sortie sur une imprimante par exemple).

Ce n'est pas le cas pour les éléments de nature ETIQUETTE qui sont obligatoirement de statut AUXILIAIRE et qui font l'objet d'une définition en phrase de type 5.

Les règles générales pour les conversions entre objets simples fondamentaux sont développées au paragraphe (2.8.3). Aussi nous contenterons nous ici d'attirer l'attention sur quelques cas spéciaux.

Le premier est celui des éléments de nature ADLIB.

Mettre en parallèle d'un objet avec un élément de nature ADLIB n'est pas à proprement parler opérer une conversion puisqu'il s'agit uniquement de déplacer de l'information sans aucunement l'altérer. Lorsqu'on fait usage de ce type d'élément en parallèle avec un autre, il lui faut prendre garde aux opérations qu'il fait subir à l'élément de nature ADLIB, ou à l'objet simple fondamental ainsi obtenu.

Comme dans toute instruction de conversion, il est nécessaire de satisfaire des contraintes portant sur la longueur des éléments

La mise en parallèle d'un élément de nature PARTITIØN et d'un élément ADLIB (les longueurs étant supposées égales) n'est légitime que si tous les objets de la partition sont des objets simples fondamentaux. Le passage se fait dans les mêmes conditions dans le sens :

PARTITIØN $\longrightarrow$ ADLIB

ou dans le sens :

ADLIB $\longrightarrow$ PARTITIØN

Comme précédemment, aucune garantie de signification n'est donnée pour le traitement ultérieur.

Les éléments de nature SYNCHRØ peuvent être mis en parallèle avec deux types d'éléments seulement :

- SYNCHRØ
- RIEN

En conséquence, si le premier élément d'une partition décrite en parallèle est de nature SYNCHRO, le premier élément de l'autre partition ne peut être que SYNCHRO ou RIEN.

Lorsqu'un élément de nature BLANC ou ZERO est mis en parallèle avec un élément de nature PARTITION tous les éléments de cette partition doivent être de nature BLANC, ZERO, MOT ou ADLIB. Ceci permet d'initialiser des zones à des valeurs ' ' ou ' 0 ', de transformer des zones de valeurs ' ' en zones de valeurs ' 0 ' et réciproquement.

En mettant en parallèle un élément de nature PARTITION et un élément de nature BOOLEEN on peut, sous certaines conditions, réaliser le "regroupement" ou "l'éclatement" de nombres booléens.

Il est nécessaire, dans ce cas, que les éléments de la partition citée comme élément parallèle soient tous de nature NATUREL ou BOOLEEN.

De la même façon la mise en parallèle d'un élément de nature PARTITION avec un élément de nature MOT permet de réaliser un ensemble d'opérations de conversion.

Il est nécessaire, dans ce cas, que tous les éléments de la partition citée soient susceptibles d'être mis en parallèle avec des éléments de nature MOT.

Un élément de nature PARTITION ne peut être mis en parallèle avec un élément de nature PROCEDURE que si les éléments de la partition citée sont de nature MOT. Cette opération est valable uniquement dans le sens :

PROCEDURE $\longrightarrow$ PARTITION

Les mises en parallèle d'éléments de nature INDEX et d'éléments de nature MOT ne sont possibles que si les conditions qui s'appliquent à ces types d'instructions de conversion sont satisfaites

Dans une conversion INDEX $\longrightarrow$ MOT, la valeur de l'index est arbitrairement prise par rapport à PREMIER (si possible).

Les conditions précédentes s'appliquent sous cette forme aux éléments simples, pour les éléments répétés et/ou conditionnels certaines précisions doivent être apportées

2.4.4.1 - ELEMENTS REPETES 1ère FORME

Plusieurs cas se présentent :

Soit ELREP1 un élément répété N fois (répétition de 1ère forme).

Il peut être mis en parallèle avec :

1. un élément répété de 1ère forme, pourvu que les facteurs de répétition aient la même valeur et que les longueurs soient identiques.

Toutefois il est possible sur des éléments de nature MØT ou BØØLEEN de ne satisfaire que la condition portant sur la longueur totale. Cela permet de réaliser des "regroupements" ou des "éclatements"

2. un élément répété de 2ème forme, pourvu que la longueur maximale de la répétition soit identique à N, ou encore que les longueurs totales des deux éléments (longueur élémentaire * longueur maximale de la répétition et longueur élémentaire * N) soient égales. Ceci permet de réaliser le "regroupement" ou "l'éclatement" d'objets de nature MØT.
3. un élément simple à condition que sa longueur soit identique à la longueur totale de la répétition.

La nature de cet élément simple ne peut être que l'une des suivantes :

BØØLEEN

MØT

RIEN

BLANC

ZERØ

ADLIB

(le type de l'élément mis en parallèle doit respecter les conditions exprimées dans le tableau du paragraphe 2.4 4)

2.4.4.2 - ELEMENTS REPETES 2ème FORME

Soit ELREP2 un élément répété de 2ème forme, N la longueur maximale de la répétition et D le drapeau.

Cet élément est obligatoirement de nature MØT.

Il peut être mis en parallèle avec :

1. un élément répété 1ère forme, à condition que les longueurs élémentaires et totales soient identiques.

Seuls sont effectivement transférés les éléments placés avant le drapeau.

2. un élément répété 2ème forme, à condition que les longueurs élémentaires et totales soient identiques.

Seuls les éléments situés avant l'occurrence du drapeau dans l'objet groupé "émetteur" peuvent être transférés, et ils le sont en totalité si le nouveau drapeau n'a pas d'occurrence parmi eux

Cela permet d'associer plusieurs drapeaux à une même séquence de caractères.

3. un élément simple, à condition que sa longueur soit identique à la longueur totale de la répétition.

2.4.4.3 - ELEMENTS DOUBLEMENT REPETES

Les conditions que doit remplir l'élément mis en parallèle avec ce type d'élément se déduisent des deux ensembles de conditions formulées ci-dessus

2.4.4.4 - ELEMENTS CONDITIONNELS

Un élément conditionnel ne peut être mis en parallèle qu'avec un élément conditionnel. Il est nécessaire que les deux éléments mis en parallèle aient des conditions similaires, de telle sorte que les occurrences soient simultanées.

2.4.4.5 - ELEMENTS COMBINES

Les conditions que doit satisfaire ce type d'élément lorsqu'on le met en parallèle avec d'autres se déduisent des conditions ci-dessus

2.4.4.6 - FORMAT DES PHRASES DE TYPE 4

Le format des phrases de type 4 est une extension de celui des phrases de type 3, sa forme générale est :

$$\langle \text{partie 1} \rangle \& \langle \text{partie 2} \rangle \left[\{ \& \langle \text{partie 10} \rangle \& \langle \text{partie 20} \rangle \} \dots \right];$$

avec :

1. < partie 1 > défini par :

< identificateur 1 > & [< condition >] & [< drapeau >] & [< nb fois >] & < nature > &
[< longueur >] & [< identificateur >]

2. < partie 2 > défini par :

< identificateur 2 > & [< condition >] & [< drapeau >] & [< nb fois >] & < nature > &
[< longueur >] & [< identificateur >]

où < identificateur 1 > et < identificateur 2 > sont les noms des partitions mises en parallèle ;

3. < partie 10 > défini par :

& [< condition >] & [< drapeau >] & [< nb fois >] & < nature > & [< longueur >] &
[< identificateur >]

4. < partie 20 > défini de la même manière.

L'élément décrit par < partie 10 > appartient à la partition désignée par < identificateur 1 > et se trouve mis en parallèle avec l'élément décrit par < partie 20 > qui appartient à la partition désignée par < identificateur 2 >.

Ainsi,

Soit ART1 et ART2 deux articles structurés respectivement par les partitions PART1 et PART2 :

DONNEE & ARTICLE & ART1 && PART1 &&& ECARTE ;

AUXILIAIRE & ARTICLE & ART2 && PART2 &&& MEMOIRE ;

PART1 & & & & MOT & 10 & NOM &

& & & & MOT & 5 & MATRICULE &

& & & & 5 & MOT & 5 & CARACT &

& & & & MOT & 10 & SALAIRE &

& & & & MOT & 6 & SECUSOC &

& & & & MOT & 24 & COMMENTAIRE ;


```

PART2      & · & & & & MØT      &   10 & NØM1 &
           & & & & & NATUREL & 6 & MATRICULE1 &
           & & & & 5 & ENTIER   & 6 & CARACT1 &
           & & & & & DECIMAL & 8.2 & SALAIRE1 &
           & & & & & DECIMAL & 4.2 & SECUSØC1 ;

```

ART1 décrit le contenu d'une carte perforée, ART2 la structure de l'article lorsqu'il a été introduit en mémoire.

Si on s'en tient à ces descriptions, on est obligé de créer un nouvel article de support MEMØIRE et dont tous les éléments sont de nature MØT pour réaliser ultérieurement des conversions entre les éléments de cet article et ceux de ART2.

On peut résumer tout cela en décrivant en parallèle PART1 et PART2.

```

PART2 & & & & MØT      &   10 & NØM & PART1 & & & & MØT & 10 & NØM &
      & & & & NATUREL & 6 & MATRICULE1 & & & & MØT & 5 & MATRICULE &
      & & & 5 & ENTIER  & 6 & CARACT1 & & & & 5 & MØT & 5 & CARACT & -
      & & & & DECIMAL & 8.2 & SALAIRE1 & & & & MØT & 10 & SALAIRE &
      & & & & DECIMAL & 4.2 & SECUSØC1 & & & & MØT & 6 & SECUSØC &
      & & & & RIEN      &      &      & & & & MØT & 24 & CØMMENTAIRE ;

```

Avec le principe adopté pour les phrases de type 4, il n'est pas possible de décrire une partition comme étant en parallèle avec plusieurs autres. Ceci est réalisé par une macro-instruction du compilateur (2.10.1.2)

2.4.5 - Sous-partition

Comme on l'a dit plus haut, une sous partition est une partition qui regroupe certains éléments de la partition dont elle est une partie. Ainsi :

Soit la partition :

```

PART1 & & & & MØT      &   15 & A1 &
      & & & & BØØLEEN & 8 & B1 &
      & & & & PARTITØN & CATAL & P1 &
      & & & & REEL      &   2.17 & R1 &
      & & & & MØT      &   15. & F1 &

```

Dans une phrase de type 2, SPART1 a fait l'objet de la description :

AUXILIAIRE & PARTITION & SPART1 & & PART1 & ;

Une phrase de type 3 a ensuite précisé sa décomposition :

SPART2 & & & & BOLEEN & 8 & B1 &
& & & & REEL & 2.17 & R1 ;

Cette description est en fait une définition puisque SPART1 n'est qu'une partition d'éléments de PART1.

Lorsqu'une phrase de type 3 est utilisée pour définir une sous partition, le format général de ces phrases introduit une limitation :

1. Tous les identificateurs cités dans la zone "nom" ne doivent avoir eu qu'une seule occurrence dans la partition dont est issue la sous-partition en cours de définition. En effet, il serait impossible, dans le cas contraire, de reconnaître l'élément auquel il est fait référence.

Cette contrainte est due au fait que les expressions qualifiées ne sont pas autorisées dans la zone "nom" des phrases de type 3.

2. La "description" de chacun des éléments d'une sous-partition doit être identique à celle de chacun des éléments référencés de la partition.
3. Une partition possédant une sous-partition ne peut structurer qu'un seul article ou qu'une seule file..

En effet, une sous-partition structure un article ou une file fictifs puisque ses éléments sont en fait les éléments de l'article ou de la file que structure la partition.

Si cette partition structurait plusieurs objets, il ne serait pas possible de savoir à quels objets se réfèrent les articles ou files structurés par la sous-partition. e.g.

AUXILIAIRE & ARTICLE & ALPHA & & B & ;

AUXILIAIRE & ARTICLE & GAMMA & & C & ;

AUXILIAIRE & ARTICLE & DELTA & & C & ;

AUXILIAIRE & PARTITION & C & & B & & ;

B & & & & NATUREL & 6 & NAT &

& & & & ENTIER & 6 & ENT &

& & & & REEL & 2,7 & REE &

& & & & MØT & 4 & MØ ;

C & & & & ENTIER & 6 & ENT &

& & & & MØT & 4 & MØ ;

Soit la représentation en mémoire de l'article ALPHA :

N A T	ENT	R E E	M Ø
-------	-----	-------	-----

Les articles GAMMA et DELTA comportent deux éléments chacun :

ENT

M Ø

Ils ont été tracés en traits interrompus du fait ils ne sont pas réellement représentés en mémoire, plus précisément :

Les éléments ENT de l'article GAMMA et ENT de l'article DELTA désignent le même objet que ENT de l'article ALPHA ; de même, les éléments MØ de l'article GAMMA et MØ de l'article DELTA désignent le même objet que MØ de l'article ALPHA.

Cette opération permet d'attribuer plusieurs noms à la même zone de mémoire.

Certaines conditions d'appliquent aux sous-partitions :

1. une même partition peut posséder plusieurs sous-partitions, mais elle ne peut, dans ce cas, structurer qu'un seul article ou qu'une seule file.
2. une sous-partition peut être définie relativement à une partition utilisée dans la description d'une partition, dans ce cas il est nécessaire que cette dernière ne structure qu'un seul article ou une seule file et que la partition sur laquelle est définie la sous-partition n'apparaisse qu'une seule fois dans les descriptions du programme
3. une sous-partition peut être utilisée pour structurer des articles ou des files, il importe alors :
 - que la nature des objets structurés par la sous-partition soit identique à celle de l'objet que structure la partition sur laquelle est définie la sous-partition (i.e. ARTICLE ou FILE). Dans le cas des natures de type FILE, il est nécessaire que le nombre d'entrées et la longueur de chaque entrée soient identiques.
 - que les statuts des objets structurés par la sous-partition satisfassent les relations suivantes, vis à vis du statut de l'objet structuré par la partition :

Statut de l'objet structuré par la partition	Statuts des objets structurés par la sous-partition
DØNNEE	DØNNEE
MIXTE	MIXTE, AUXILIAIRE
RESULTAT	RESULTAT
AUXILIAIRE	AUXILIAIRE

Les places des objets doivent être identiques

* REMARQUE :

La correspondance, par l'intermédiaire d'une sous-partition, d'un objet de statut MIXTE et d'objets de statut AUXILIAIRE revient à réaliser une procédure incomplète. L'utilisation de cette possibilité conduit à une économie de place, mais demande une grande attention quand à son emploi.

4. Dans le cas où une sous-partition figure dans un appel de procédure, elle doit figurer parmi les arguments "donnée".

Elle figure obligatoirement dans l'appel de procédure lorsque un ou plusieurs objets qu'elle "structure" figurent dans l'appel de procédure. Elle doit être décrite par une phrase de type 3 (ou 4) ainsi que la partition dont elle est issue.

Il est aussi possible de décrire une sous-partition par phrase de type 4, dans ce cas la partition avec laquelle elle est mise en parallèle peut être une autre sous-partition.

Une autre technique peut être utilisée pour la définition d'une sous-partition, elle consiste à utiliser une phrase de type 4 décrivant la partition et sa sous-partition. Ceci permet de changer les noms des éléments regroupés dans la sous-partition.

Les éléments de la partition pour lesquels il n'est pas désiré d'occurrence dans la sous-partition sont mis en parallèle avec des éléments de nature RIEN. Les éléments mis en parallèle doivent respecter les règles précédemment énoncées.

2.4.6 - Partitions cataloguées

Rappelons que l'utilisateur peut créer un catalogue de partitions où il fait figurer les descriptions des partitions qu'il utilise, cela lui évite d'avoir à les décrire à nouveau dans chaque programme.

Une partition est entrée dans le catalogue sous deux formes, la forme externe (celle-là même sous laquelle le programmeur l'a écrite) et une forme codée utilisée par le compilateur.

Une partition est toujours ajoutée seule au catalogue, même si elle a été décrite en parallèle ou comme une sous-partition : les partitions qu'elle utilise pour sa description ne sont pas cataloguées avec elle. Cela explique qu'on puisse mentionner CATAL dans la zone longueur des phrases de type 3 ou 4.

La façon dont ce catalogue est géré n'est pas encore figée. On tiendra compte du système d'exploitation et des goûts des usagers.

D'ores et déjà on peut envisager une organisation à trois niveaux, qui conduit à créer trois types de catalogues :

- des catalogues privés, un par utilisateur
- des catalogues semi-privés, avec numéro de compte
- un catalogue standard.

Ces catalogues seront gérés à l'aide de deux macro-instructions (2.10.1.3) qui permettent respectivement d'ajouter et de retrancher une partition à un catalogue.

2.5 - DESIGNATION DES OBJETS

Les éléments d'une partition sont désignés par des expressions qualifiées.

Elles sont de la forme suivante :

Partition	Condition	Drapeau	Nb fois	Nature	Longueur	Nom	Désignation
A				*	*	B	$B \square DE \square A$
			.	.	.	C	$C(i) \square DE \square A$
			.	MØT	*	D	$D(j) \square DE \square A$
			.	MØT	.	E ;	$E(i)(j) \square DE \square A$
A1	α			*	*	B1	$B1(\alpha) \square DE \square A1$
	α		*	*	*	C1	$C1(i)(\alpha) \square DE \square A1$
	α	*		MØT	*	D1	$D1(j)(\alpha) \square DE \square A1$
	α	*	*	MØT	.	E1 ;	$E1(i)(j)(\alpha) \square DE \square A1$

Les mots $B \square DE \square A$, _____, $E1(i)(j)(\alpha) \square DE \square A1$ sont des expressions qualifiées.

Elles sont construites en examinant la phrase de type 3 ou 4, qui a décrit la partition, de la droite vers la gauche et en précisant au passage tous les éléments nécessaires à l'identification de l'objet qu'on veut désigner.

Les paragraphes suivants précisent cette règle générale (2.5.1 à 2.5.4).

2.5.1 - Désignation des éléments dans les partitions

2.5.1.1 - ELEMENT SIMPLE

On désigne un élément simple d'une partition par l'expression qualifiée :

$\langle \text{nom de l'élément} \rangle \square DE \square \langle \text{nom de partition} \rangle$

C'est la raison pour laquelle deux éléments ne peuvent porter le même nom dans la même phrase de type 3 (ou dans la même partie de phrase de type 4) : il serait impossible de les distinguer.

2.5.1.2 - ELEMENT REPETE 1ère FORME

La désignation d'un élément répété 1ère forme dans la partition ou il est décrit est faite par l'expression qualifiée (voir 2.5.1.1) :

$\langle \text{nom de l'élément} \rangle \left[(\langle \text{indice} \rangle) \right] \square \text{ DE } \square \langle \text{nom de partition} \rangle$

$\langle \text{indice} \rangle$ étant soit une constante de nombre naturel soit une variable désignant un objet de nature NATUREL

$\langle \text{indice} \rangle$ permet d'identifier un élément dans la répétition. Ainsi, avec la description suivante :

PARTITION 1 & _____ &
 _____ &
 & & 15 & ENTIER & 6 & INTEG &
 _____ ;

les expressions qualifiées

INTEG (1) $\square$ DE $\square$ PARTITION 1
 INTEG (2) $\square$ DE $\square$ PARTITION 1
 et INTEG (15) $\square$ DE $\square$ PARTITION 1

désignent respectivement :

le premier, le second, le quinzième et dernier élément de la répétition

Les expressions qualifiées :

INTEG (0) $\square$ DE $\square$ PARTITION 1
 et INTEG (16) $\square$ DE $\square$ PARTITION 1

sont dépourvues de toute signification puisque la valeur de $\langle \text{indice} \rangle$ sort des limites permises (depuis, 1 par définition, jusqu'à la valeur définie par le facteur de répétition, 15 ici).

L'expression qualifiée

INTEG □ DE □ PARTITION 1

désigne la suite ordonnée des éléments répétés INTEG, considérée comme un tout, i.e.

INTEG (1)	}	□ DE □ PARTITION 1
INTEG (2)		
.		
INTEG (15)		

2.5.1.3 - ELEMENT REPETE 2ème FORME

La désignation d'un élément répété 2ème forme dans la partition où il est décrit est faite par l'expression qualifiée (voir 2.5.1.1) :

< nom de l'élément > [(< indice >)] □ DE □ < nom de partition >

< indice > étant soit une constante de nombre naturel soit une variable désignant un objet de nature NATUREL.

< indice > permet de repérer un élément dans la répétition ; ainsi avec la description suivante :

PARTITION 2 & _____ &
 & _____ &
 & & ' / * " , 20 & & MØT & 2 & TEXTE &
 & _____
 _____ ;

les expressions qualifiées

TEXTE (1) □ DE □ PARTITION 2

TEXTE (2) □ DE □ PARTITION 2

et TEXTE (20) □ DE □ PARTITION 2

désignent, sous réserve de signification (voir ci-après), respectivement :

le premier, le second, le 20ème et dernier élément de la répétition.

Les expressions qualifiées :

TEXTE (0) □ DE □ PARTITION 2
et TEXTE (J) □ DE □ PARTITION 2

sont dépourvues de signification :

- la 1ère du fait que la répétition commence à la valeur d'indice 1 ;
- la seconde dans deux cas :
 - la valeur désignée par J est égale à 0 ou supérieure à la longueur maximale de la répétition (ici 20)
 - l'élément désigné par l'expression qualifiée est situé après l'apparition du drapeau.

D'une façon plus précise, avec l'exemple précédent, si les valeurs des éléments répétés TEXTE sont :

TEXTE (1) = ' AB "

TEXTE (2) = ' CD "

TEXTE (3) = ' EF "

TEXTE (4) = ' GH "

TEXTE (5) = ' / * ."

l'expression qualifiée :

TEXTE (J) □ DE □ PARTITION 2

est dépourvue de signification si J désigne un nombre naturel supérieur ou égal à 5. Dans ce cas les éléments répétés sont :

TEXTE (1) □ DE □ PARTITION 2

TEXTE (2) □ DE □ PARTITION 2

TEXTE (3) □ DE □ PARTITION 2

TEXTE (4) □ DE □ PARTITION 2

Pour revenir à l'exemple général, l'expression qualifiée :

TEXTE □ DE □ PARTITION 2

désigne la suite ordonnée des éléments répétés TEXTE depuis le premier jusqu'au dernier élément - celui qui précède le drapeau - ; en conséquence cette expression désigne, au maximum si le drapeau n'est pas apparu, l'ensemble des éléments :

TEXTE (1)	}	□ DE □ PARTITION 2
TEXTE (2)		
...		
TEXTE (20)		

Il est possible, à l'aide de moyens appropriés (2.7.10.2.2), de connaître le nombre d'occurrences d'un élément répété 2ème forme.

2.5.1.4 - ELEMENT DOUBLEMENT REPETE

La désignation d'un élément doublement répété dans la partition où il est décrit est faite par l'expression qualifiée (voir 2.5.1.1) :

< nom de l'élément > $\left[\left[(< \text{indice 1} >) \right] (< \text{indice 2} >) \right] \square \text{ DE } \square < \text{nom de partition} >$

< indice 1 > et < indice 2 > étant soit des constantes de nombres naturels soit des variables désignant des objets de nature NATUREL.

< indice 1 > permet de repérer un élément dans la répétition de 1ère forme et < indice 2 > un élément dans celle de 2ème forme. Par exemple, soit la description suivante :

PARTITION 3 & _____ &

& & ' // ', 30 & 2 & MÔT & 2 & COMMENT

les expressions qualifiées

COMMENT (1) (1) □ DE □ PARTITION 3

COMMENT (2) (1) □ DE □ PARTITION 3

COMMENT (1) (2) □ DE □ PARTITION 3

sous réserve qu'elles aient une signification, désignent respectivement :

- le 1^{er} élément de la 1^{ère} répétition du type drapeau
- le 2^{ème} élément de la 1^{ère} répétition du type drapeau
- le 1^{er} élément de la 2^{ème} répétition du type drapeau

L'expression qualifiée :

~~C~~OMMENT (I) (J) □ DE □ PARTIT~~I~~ØN 3

est dépourvue de signification si :

- I désigne un nombre naturel inférieur à 1 ou supérieur à 2 (limites de la répétition de 1^{ère} forme)
- où bien J désigne un nombre égal à 0⁰ ou supérieur à 30 ou bien, avec $I \in \{1, 2\}$ et $J \in \{1, 30\}$, l'élément désigné ne précède pas l'occurrence du drapeau
'// "

L'expression qualifiée :

~~C~~OMMENT (K) □ DE □ PARTIT~~I~~ØN 3

désigne la suite ordonnée des éléments auxquels feraient référence les expressions qualifiées :

~~C~~OMMENT (1) (K) □ DE □ PARTIT~~I~~ØN 3

~~C~~OMMENT (2) (K) □ DE □ PARTIT~~I~~ØN 3

dans la mesure où K remplit les conditions énumérées ci-dessus. En d'autres termes :

Pour un élément doublement répété l'omission de l'un des deux indices entraîne que l'indice restant est considéré comme celui relatif à la répétition du type < drapeau > (2^{ème} forme).

L'expression qualifiée :

~~C~~OMMENT □ DE □ PARTIT~~I~~ØN 3

désigne la suite ordonnée des éléments qui apparaissent jusqu'à l'occurrence du drapeau (celui-ci non compris).

2.5.1.5 - ELEMENT CONDITIONNEL

La désignation d'un élément conditionnel dans la partition où il est décrit est faite par l'expression qualifiée (voir 2.5.1.1) :

< nom de l'élément > (< condition >) □ DE □ < nom de partition >

< condition > étant une expression de condition identique à celle qui figure devant l'élément dans la description de la partition (ou une condition englobant celle-ci).

Soit par exemple la description de partition :

PARTITION 5 & & & NATUREL & 4 & CØND 1 &
 & & & NATUREL & 4 & CØND 2 &
 & & & BØØLEEN & 8 & TEST &
 & CØND 1 ≥ CØND 2 & & MØT & 20 & CARACT &
 & CØND 2 = CØND 1 & & ENTIER & 6 & NUMERØ &
 & TEST □ DE □ PARTITION 5 & & REEL & 2,7 & FLØAT &

Les éléments CARACT, NUMERØ et FLØAT sont désignés respectivement par les expressions qualifiées :

CARACT (CØND 1 ≥ CØND 2) □ DE □ PARTITION 5

NUMERØ (CØND 1 = CØND 2) □ DE □ PARTITION 5

FLØAT (TEST □ DE □ PARTITION 5) □ DE □ PARTITION 5

REMARQUE :

En toute rigueur les deux premières expressions qualifiées (ainsi que l'expression des conditions dans la description de la partition) devraient s'écrire :

CARACT (CØND 1 □ DE □ PARTITION 5 ≥ CØND 2 □ DE □ PARTITION 5) □ DE □ PARTITION 5

NUMERØ (CØND 1 □ DE □ PARTITION 5 = CØND 2 □ DE □ PARTITION 5) □ DE □ PARTITION 5

Il est possible de modifier l'expression de la condition pour les éléments ayant une condition du type :

< variable > < symbole de comparaison > < nombre >

les deux autres types de condition :

< variable > < symbole de comparaison > < variable >

et < variable >

doivent avoir la même forme dans la description de la partition et dans l'expression qualifiée désignant l'élément conditionnel. Soit la description de partition :

PARTITION 4 & & & NATUREL & 2 & K &
 & K > 0 & & & MÔT & 10 & PÔSITIF &
 & & & & NATUREL & 2 & NUMERÔ &
 & K = 1 & & & MÔT & 10 & EGAL 1 &
 & K = 2 & & & DECIMAL & 10.2 & EGAL 2 ;

les expressions qualifiées :

PÔSITIF (K > 0) □ DE □ PARTITION 4

EGAL 1 (K = 1) □ DE □ PARTITION 4

EGAL 2 (K = 2) □ DE □ PARTITION 4

désignent respectivement

- l'élément PÔSITIF de PARTITION 4

- l'élément EGAL 1 de PARTITION 4

- l'élément EGAL 2 de PARTITION 4

La condition "K > 0" est équivalente, ici à la condition "K = 1 ou K = 2", de sorte que l'élément PÔSITIF de PARTITION 4 peut être désigné par l'une ou l'autre des expressions qualifiées :

PÔSITIF (K = 1) □ DE □ PARTITION 4

PÔSITIF (K = 2) □ DE □ PARTITION 4

par contre les expressions qualifiées :

EGAL 1 (K > 0) DE PARTITION 4

EGAL 2 (K > 0) DE PARTITION 4

n'ont aucune signification puisque EGAL 1 n'est présent que si K = 1 et EGAL 2 que si K = 2.

REMARQUE IMPORTANTE :

Une expression qualifiée désignant un élément conditionnel n'a de signification que si la condition, dont elle contient l'expression, est satisfaite. Dans le cas contraire, une erreur est détectée à l'exécution.

2.5.1.6 - ELEMENT COMBINE

Plusieurs cas se présentent, mais les règles à satisfaire pour chacun d'eux sont les règles qui ont été précédemment énoncées pour les éléments (conditionnels, répétés 1ère forme, répétés 2ème forme) à partir desquels la construction de l'élément combiné se déduit

2.5.1.6.1 - ELEMENT CONDITIONNEL + REPETITION DE 1ère FORME

La désignation d'un tel élément dans la partition où il est décrit est faite par l'expression qualifiée (voir 2.5.1.1) :

< nom de l'élément > [(< indice >)] (< condition >) □ DE □ < nom de partition >

< indice > ayant la même définition qu'en 5.1.2 et < condition > la même définition qu'en 5.1.5.

Soit la description de partition :

PARTITION 6 & · & & NATUREL & 2 & K1 &

& & & NATUREL & 2 & K2 &

& & & BØØLEEN & 8 & BØØL &

& K1 > K2 & & K1 & DECIMAL & 6.2 & DEC1 &
 & K1 = K2 & & K2 & DECIMAL & 6.2 & DEC2 &
 & K1 < K2 & & 30 & DECIMAL & 6.3 & DEC3 &
 & BØØL & & 10 & MØT & 10 & CUMMUL ;

les expressions qualifiées

DEC1 (I) (K1 > K2) □ DE □ PARTITION 6

DEC2 (J) (K1 = K2) □ DE □ PARTITION 6

DEC3 (N) (K1 < K2) □ DE □ PARTITION 6

et CUMMUL (5) (BØØL) □ DE □ PARTITION 6

désignent respectivement :

- le I^{ème} élément répété de DEC1, sous la condition K1 > K2,
- le J^{ème} élément répété de DEC2, sous la condition K1 = K2,
- le N^{ème} élément répété de DEC3, sous la condition K1 < K2,
- le 5^{ème} élément répété de CUMMUL sous la condition BØØL

Pour que ces expressions qualifiées aient une signification, il faut que :

- la condition exprimée soit satisfaite ;
- la valeur de < indice > (un nombre naturel) ne soit ni égale à 0 ni supérieure au facteur de répétition .

Ces conditions ont été exprimées aux paragraphes 5.1.2 et 5.1.5.

Une expression qualifiée du type :

< nom de l'élément > (< condition >) □ DE □ < nom de partition >

pour désigner un élément conditionnel avec répétition de 1^{ère} forme, désigne l'ensemble des éléments répétés, e.g. dans l'exemple précédent.

DEC3 (K1 < K2) □ DE □ PARTITION 6

désigne l'ensemble des éléments :

DEC3 (1) ($K1 < K2$) □ DE □ PARTITION6

DEC3 (2) ($K1 < K2$) □ DE □ PARTITION6

.....

DEC3 (30) ($K1 < K2$) □ DE □ PARTITION6

2.5.1.6.2 - ELEMENT CONDITIONNEL + REPETITION DE 2^{ème} FORME

La désignation d'un tel élément dans la partition où il est décrit est faite par l'expression qualifiée (voir 2.5.1.1) :

< nom de l'élément > [(< indice >)] (< condition >) □ DE □ < nom de partition >

< indice > ayant la même définition qu'en 5.1.5.

Soit la description de partition :

PARTITION7 & & & NATUREL & 2 & KZ &

& & & BØØLEEN & 8 & BØØL &

& KZ = 0 & ' * / ', 80 & & MØT & 2 & CØMF &

& KZ = 1 & ' / * ', 80 & & MØT & 2 & CØMD &

& BØØL & ' := ', 80 & & MØT & 2 & AFFECT ;

les expressions qualifiées :

CØMF (ØN) ($KZ = 0$) □ DE □ PARTITION7

CØMD (BN) ($KZ = 1$) □ DE □ PARTITION7

AFFECT (15) (BØØL) □ DE □ PARTITION7

désignent respectivement :

- le ØN^{ième} élément répété de CØMF, sous la condition $KZ = 0$,
- le BN^{ième} élément répété de CØMD, sous la condition $KZ = 1$,
- le 15^{ième} élément répété de AFFECT, sous la condition BØØL

Pour que ces expressions aient une signification il faut que :

- la condition exprimée soit remplie,
- la valeur de $\langle \text{indice} \rangle$ (un nombre naturel) soit supérieure à 0 et au plus égale à celle qui correspond au dernier élément (soit avant l'occurrence du drapeau ou à la limite de la répétition).

Ces conditions ont été précisées aux paragraphes 2.5.1.3 et 2.5.1.5.

Une expression qualifiée du type :

$\langle \text{nom de l'élément} \rangle (\langle \text{condition} \rangle) \square \text{DE} \square \langle \text{nom de partition} \rangle$

pour désigner un élément conditionnel avec répétition de 2ème forme, désigne l'ensemble des éléments répétés jusqu'à l'occurrence du drapeau (non compris) ou jusqu'à la longueur maximale dans le cas contraire.

Ainsi, dans l'exemple précédent :

$\text{AFFECT}(\text{B}\emptyset\emptyset\text{L}) \square \text{DE} \square \text{PARTITION}7$

désigne l'ensemble des éléments :

$\text{AFFECT}(1)(\text{B}\emptyset\emptyset\text{L}) \square \text{DE} \square \text{PARTITION}7$

$\text{AFFECT}(2)(\text{B}\emptyset\emptyset\text{L}) \square \text{DE} \square \text{PARTITION}7$

$\text{AFFECT}(\text{MAX})(\text{B}\emptyset\emptyset\text{L}) \square \text{DE} \square \text{PARTITION}7$

MAX étant l'indice du dernier élément avant l'occurrence du drapeau ou, si celui-ci n'apparaît pas, le nombre maximal de répétitions

Dans l'exemple ci-dessus, on a supposé que $\text{MAX} \geq 3$.

2.5.1.6.3 - ELEMENT CONDITIONNEL + DOUBLE REPETITION

La désignation d'un tel élément dans la partition où il est décrit est faite de l'expression qualifiée (voir 2.5.1.1) :

$\langle \text{nom de l'élément} \rangle \left[\left[(\langle \text{indice 1} \rangle) \right] (\langle \text{indice 2} \rangle) \right] (\langle \text{condition} \rangle) \square \text{DE} \square$
 $\langle \text{nom de partition} \rangle$

< indice 1 > est relatif à la répétition de 1ère forme (2.5.1.2).

< indice 2 > à celle de 2ème forme (2.5.1.3) et < condition > a la même définition qu'en 2.5.1.5.

Soit la description de partition :

```

PARTITION8  & & & ENTIER  & 4 & ENT &
             & & & B00LEEN & 8 & BIN &
             & ENT = - 2 & ' *"', 30 & 10 & M0T & 1 & SUITE 1 &
             & ENT = 4  & ' / "', 20 & 10 & M0T & 1 & SUITE 2 &
             & BIN  & ' *"', 40 & 10 & M0T & 2 & EXP0N  ;
  
```

les expressions qualifiées :

SUITE 1 (I) (J) (ENT = -2) □ DE □ PARTITION8

SUITE 2 (K) (L) (ENT = 4) □ DE □ PARTITION8

et EXP0N (M) (N) (BIN) □ DE □ PARTITION8

désignent respectivement :

- le I^{ème} élément de SUITE1 de la J^{ème} répétition du groupe de 10 éléments, sous la condition ENT = -2
- le K^{ème} élément de SUITE2 de la L^{ème} répétition du groupe de 10 éléments, sous la condition ENT = 4
- le M^{ème} élément de EXP0N de la N^{ème} répétition du groupe de 10 éléments, sous la condition BIN.

Pour que ces expressions qualifiées aient une signification, il faut que :

- la condition exprimée soit remplie,
- la valeur de < indice 1 > (un nombre naturel) soit supérieure à zéro et au plus égale au nombre de répétition de 1ère forme (< Nb fois >),
- la valeur de < indice 2 > (un nombre naturel) soit supérieure à zéro et inférieure ou égale soit :
 - à l'indice du dernier élément précédant l'occurrence du drapeau,
 - au nombre maximal de répétition selon le cas.

Ces conditions ont été précisées aux paragraphes 2.5.1.2, 2.5.1.3, 2.5.1.4 et 2.5.1.5.

Les expressions qualifiées de l'une des deux formes :

< nom de l'élément > (< indice >) (< condition >) □ DE □ < nom de partition >

ou < nom de l'élément > (< condition >) □ DE □ < nom de partition >

pour désigner un élément conditionnel avec double répétition, désignent respectivement ici :

1er Cas :

L'ensemble des éléments répétés 1ère forme (< Nb fois >) pour la valeur < indice > relative à la répétition du type drapeau. e.g. :

dans l'exemple précédent, l'expression qualifiée :

SUITE 2 (3) (ENT = 4) □ DE □ PARTITION 8

désigne l'ensemble des éléments :

SUITE 2 (1) (3) (ENT = 4) □ DE □ PARTITION 8

SUITE 2 (2) (3) (ENT = 4) □ DE □ PARTITION 8

.....

SUITE 2 (10) (3) (ENT = 4) □ DE □ PARTITION 8

sous réserve que les conditions ci-dessus exprimées sont satisfaites par < indice > et < condition >.

2ème Cas

L'ensemble des éléments doublement répétés. Ainsi dans l'exemple précédent, l'expression qualifiée :

SUITE 1 (ENT = -2) □ DE □ PARTITION 8

désigne l'ensemble des éléments :

SUITE 1 (1) (1) (ENT = -2) □ DE □ PARTITION 8

SUITE 1 (2) (1) (ENT = -2) □ DE □ PARTITION 8

.....

SUITE 1 (10) (1) (ENT = -2) □ DE □ PARTITION 8

SUITE 1 (1) (2) (ENT = -2) □ DE □ PARTITION 8

SUITE 1 (2) (2) (ENT = -2) □ DE □ PARTITION 8

.....

SUITE 1 (I) (J) (ENT = -2) □ DE □ PARTITION 8

avec soit :

- SUITE 1 (I + 1) (J) (ENT = -2) □ DE □ PARTITION 8 ou si I = 10 (< nb fois >)

SUITE 1 (1) (J + 1) (ENT = -2) □ DE □ PARTITION 8 valant ' * ' (drapeau)

- ou encore I = 10 et J = 10 si le drapeau n'a pas eu d'occurrence.

REMARQUE :

Dans la plupart des exemples précédents, la règle de formation des expressions qualifiées n'a pas été respectée dans un cas au moins : à savoir lorsqu'un élément de la partition était utilisé dans celle-ci.

Un tel élément - une condition par exemple - est employé tel que, sans référence aucune à la partition dont il fait partie.

On a pris cette liberté, répétons-le par souci de clarification.

Si on s'en tenait strictement aux règles énoncées jusqu'ici; la description de PARTITION 8, utilisée ci-dessus, devrait s'écrire :

$\text{PARTITION } 8 \quad \& \& \& \& \text{ENTIER} \quad \& \quad 4 \quad \& \text{ENT} \quad \&$
 $\& \& \& \& \text{BOOLEEN} \quad \& \quad 8 \quad \& \text{BIN} \quad \&$
 $\& \text{ENT} \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8 = -2 \quad \& \quad ' * ', 30 \quad \& \quad 10 \quad \& \text{MOT} \quad \& \quad 1 \quad \& \text{SUITE } 1 \quad \&$
 $\& \text{ENT} \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8 = 4 \quad \& \quad ' / ', 20 \quad \& \quad 10 \quad \& \text{MOT} \quad \& \quad 1 \quad \& \text{SUITE } 2 \quad \&$
 $\& \text{BIN} \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8 \quad \& \quad ' ** ', \quad 40 \quad \& \quad 10 \quad \& \text{MOT} \quad \& \quad 2 \quad \& \text{EXPON} ;$

et les premières expressions qualifiées, citées en exemple au début de ce paragraphe, devraient s'écrire :

$\text{SUITE } 1 \text{ (I) (J) (ENT} \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8 = -2) \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8$
 $\text{SUITE } 1 \text{ (K) (L) (ENT} \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8 = 4) \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8$
 et $\text{EXPON (M) (N) (BIN} \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8) \sqsubseteq \text{DE} \sqsubseteq \text{PARTITION } 8$

où I, J, K, L, M et N peuvent être des expressions qualifiées.

On peut le faire, mais l'écriture s'en trouve alourdie d'autant. Et comme elle est parfois bien lourde encore, nous allons étendre le procédé de simplification à toutes les expressions qualifiées sans distinction au paragraphe 2.5.4

Mais au paravant nous allons généraliser la notion de partition.

2.5.2 - Généralisation dans le cas des partitions

Les expressions qualifiées présentées jusqu'à présent ne portent pas sur des partitions contenant des éléments de nature PARTITION.

Dans le cas général des partitions, il importe de "remonter" l'arborescence depuis l'élément à désigner en qualifiant chacun des sommets de celle-ci, e.g.

Soit la description de partition :

GENERAL 1 & & & NATUREL & 4 & K &

& & & 10 & PARTITION & & PARTIEL 1 &

& & & PARTITION & & PARTIEL 2

& K □ DE □ GENERAL 1 = 1 & & PARTITION & & PARTIEL 1 &

& K □ DE □ GENERAL 1 = 2 & & 10 & PARTITION & & PARTIEL 2 &

& & & MOT & 20 & COMMENTAIRE ;

avec

PARTIEL 1 & & & PARTITION & & PARTIEL 2 &

& K □ DE □ GENERAL 1 = 1 & & PARTITION & & PARTIEL 2 &

& & & MOT & 20 & COMMENTAIRE ;

et

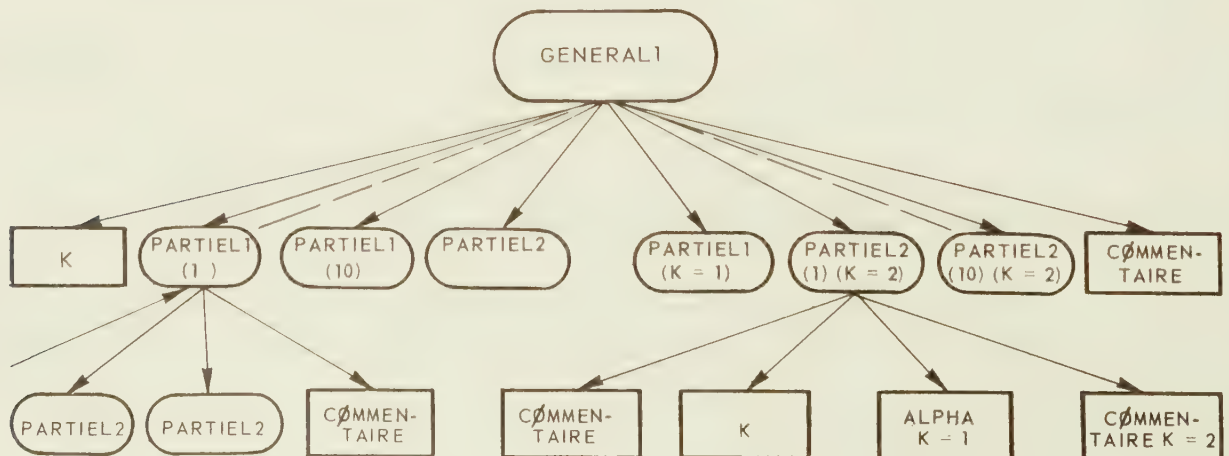
PARTIEL 2 & & & MOT & 10 & COMMENTAIRE &

& & & NATUREL & 4 & K &

& K □ DE □ GENERAL 1 = 1 & & MOT & 10 & ALPHA &

& K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2 & & MOT & 10 & COMMENTAIRE ;

La structure arborescente résultante est, en abrégé ;



Les éléments de la suite nommée ~~C~~OMMENTAIRE dans la partition GENERAL 1, sont désignés par :

~~C~~OMMENTAIRE □ DE □ GENERAL 1

~~C~~OMMENTAIRE □ DE □ PARTIEL 1 (1) □ DE □ GENERAL 1

.....

~~C~~OMMENTAIRE □ DE □ PARTIEL 1 (10) □ DE □ GENERAL 1

~~C~~OMMENTAIRE □ DE □ PARTIEL 2 □ DE □ PARTIEL 1 (1) □ DE □ GENERAL 1

....

~~C~~OMMENTAIRE □ DE □ PARTIEL 2 □ DE □ PARTIEL 1 (20) □ DE □ GENERAL 1

~~C~~OMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2 □
DE □ PARTIEL 1 (1) □ DE □ GENERAL 1

.....

~~C~~OMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2 □
DE □ PARTIEL 1 (10) □ DE □ GENERAL 1

~~C~~OMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2 □
(K □ DE □ GENERAL 1 = 1) □ DE □ PARTIEL 1 (1) □ DE □ GENERAL 1

.....

~~C~~OMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2
(K □ DE □ GENERAL 1 = 1) □ DE □ PARTIEL 1 (10) □ DE □ GENERAL 1

~~C~~OMMENTAIRE □ DE □ PARTIEL 2 (K □ DE □ GENERAL 1 = 1) □ DE □ PARTIEL 1 (1) □
DE □ GENERAL 1

.....

~~C~~OMMENTAIRE □ DE □ PARTIEL 2 (K □ DE □ GENERAL 1 = 1) □ DE □ PARTIEL 1 (10) □
DE □ GENERAL 1

COMMENTAIRE □ DE □ PARTIEL 2 □ DE □ GENERAL 1

COMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2 □
DE □ GENERAL 1

COMMENTAIRE □ DE □ PARTIEL 1 (K □ DE □ GENERAL 1 = 1) □ DE □ GENERAL 1

COMMENTAIRE □ DE □ PARTIEL 2 □ DE □ PARTIEL 1 (K □ DE □ GENERAL 1 = 1) □
DE □ GENERAL 1

COMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2 □
DE □ PARTIEL 1 (K □ DE □ GENERAL 1) □ DE □ GENERAL 1

COMMENTAIRE □ DE □ PARTIEL 2 (K □ DE □ GENERAL 1 = 1) □ DE □ PARTIEL 1
(K □ DE □ GENERAL 1 = 1) □ DE □ GENERAL 1

COMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2
(K □ DE □ GENERAL 1 = 1) □ DE □ PARTIEL 1 (K □ DE □ GENERAL 1 = 1) □ DE □
GENERAL 1

COMMENTAIRE □ DE □ PARTIEL 2 (1) (K □ DE □ GENERAL 1 = 2) □ DE □ GENERAL 1

.....

COMMENTAIRE □ DE □ PARTIEL 2 (10) (K □ DE □ GENERAL 1 = 2) □ DE □ GENERAL 1

COMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2 (1)
(K □ DE □ GENERAL 1 = 2) □ DE □ GENERAL 1

.....

COMMENTAIRE (K □ DE □ PARTIEL 2 □ DE □ GENERAL 1 = 2) □ DE □ PARTIEL 2 (10)
(K □ DE □ GENERAL 1 = 2) □ DE □ GENERAL 1

L'ensemble de ces expressions permet de désigner les 78 éléments nommés **COMMENTAIRE** dans la partition **GENERAL 1**.

Nous avons choisi à dessin cet exemple un peu compliqué, afin de mettre en évidence les possibilités offertes pour les expressions qualifiées pour désigner les objets

Dans le cas d'une expression qualifiée, dont la forme générale est :

$\langle \text{qualificateur}_0 \rangle \square \text{DE} \square \langle \text{qualificateur}_1 \rangle \dots \square \text{DE} \square \langle \text{qualificateur}_i \rangle \square \text{DE}$

Les identificateurs apparaissent dans $\langle \text{qualificateur}_i \rangle$ ($i \geq 1$) désignent les partitions, en conséquence $\langle \text{qualificateur}_i \rangle$ est ainsi défini :

$$\langle \text{identificateur}_i \rangle \left[\left[(\langle \text{indice } 1 \rangle) \right] (\langle \text{condition} \rangle) \right]$$

pour $i \geq 1$

$\langle \text{identificateur}_i \rangle$ désigne une partition. Tous les $\langle \text{identificateur}_i \rangle$, lorsque $i \geq 1$, doivent être différents les uns des autres. Ceci est une conséquence du fait que les descriptions récurrentes de partitions sont interdites

$\langle \text{qualificateur}_0 \rangle$ désigne un élément (qui peut être de nature **PARTITION**, il doit alors satisfaire à la condition ci-dessus) de la partition, il est défini par :

$$\langle \text{identificateur} \rangle \left[\left[\left[(\langle \text{indice } 1 \rangle) \right] (\langle \text{indice } 2 \rangle) \right] (\langle \text{condition} \rangle) \right]$$

2.5.3 - Emploi des expressions qualifiées

Les partitions structurent des articles ou des files

Les expressions qualifiées sont utilisées pour :

- identifier, dans la description d'une partition, un élément apparaissant dans les zones "condition", "drapeau", "nb fois" notamment
- isoler un élément ou une partie d'un article ou d'une file **placé en mémoire**.

On a examiné le premier cas lorsqu'on a défini les descriptions de partition (2.4). On va maintenant examiner le second

2.5.3.1 - ELEMENT OU PARTIE D'UN ARTICLE PLACE EN MEMOIRE

Le format général est de l'expression qualifiée qui désigne un tel objet est le suivant :

$$\langle \text{qualificateur}_0 \rangle \square \text{DE} \square \left[\{ \cdot \text{qualificateur}_1 \rangle \square \text{DE} \square \{ \dots \} \right] \langle \text{identificateur} \rangle$$

$\langle \text{identificateur} \rangle$ est le nom d'un article placé obligatoirement en mémoire. Le dernier qualificateur est, logiquement, un identificateur qui désigne la partition structurant l'article.

$\langle \text{qualificateur}_0 \rangle$ et $\langle \text{qualificateur}_i \rangle$, $i \geq 1$, ont été définis ci-dessus.

2.5.3.2 - ELEMENT OU PARTIE D'UNE FILE PLACEE EN MEMOIRE

Plusieurs formes sont possibles :

2.5.3.2.1 - ELEMENT DE FILE

La première forme est :

$$\langle \text{nom de file} \rangle (\langle \text{expqua}_1 \rangle \left[\{ , \langle \text{expqua}_2 \rangle \{ \dots \} \right])$$

où :

$\langle \text{nom de file} \rangle$ est le nom d'une file placée en mémoire.

$\langle \text{expqua}_i \rangle *$ est un identificateur ou une expression qualifiée désignant l'un des index attachés à l'entrée correspondante de la file.

Cette expression désigne un article, qui est un élément de la file, on désigne l'un des éléments de cet article, par une expression de la forme suivante :

$$\langle \text{qualificateur}_0 \rangle \square \text{DE} \square \left[\langle \text{qualificateur}_1 \rangle \square \text{DE} \square \{ \dots \} \right]$$

$$\langle \text{nom de file} \rangle (\langle \text{expqua}_1 \rangle \left[\{ , \langle \text{expqua}_2 \rangle \{ \dots \} \right])$$

En vertu des règles énoncées plus haut, le qualificateur qui précède le nom de la file est un identificateur désignant la partie qui structure cette file.

* REMARQUE :

Il est possible de subsister à $\langle \text{expqua}_i \rangle$ l'un des deux mots réservés

PREMIER

DERNIER

PREMIER et DERNIER étant respectivement la première et la dernière des positions possibles pour un index rattaché à l'entrée correspondante de la file.

2.5.3.2.2 - PARTIE DE FILE

Le format général est :

$\langle \text{nom de file} \rangle (\langle \text{index partie de file}_1 \rangle [\{ , \langle \text{index partie de file}_2 \rangle \} \dots])$

où $\langle \text{nom de file} \rangle$ est le nom d'une file placée en mémoire et $\langle \text{index partie de file}_i \rangle$ est défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \text{DE } \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \text{PREMIER} \\ \text{DERNIER} \end{array} \right\} \square \text{JUSQUA} \square \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \text{PREMIER} \\ \text{DERNIER} \end{array} \right\} \end{array} \right\}$$

L'un au moins des $\langle \text{index partie de file}_i \rangle$ n'étant pas du type $\langle \text{expqua} \rangle$

Les diverses configurations possibles de $\langle \text{index partie de file} \rangle$ sont :

$\langle \text{expqua} \rangle$

DE $\square \langle \text{expqua} \rangle \square$ JUSQUA $\square$ PREMIER

DE $\square \langle \text{expqua} \rangle \square$ JUSQUA $\square$ DERNIER

DE $\square \langle \text{expqua} \rangle \square$ JUSQUA $\square \langle \text{expqua} \rangle$

DE $\square$ PREMIER $\square$ JUSQUA $\square \langle \text{expqua} \rangle$

DE $\square$ PREMIER $\square$ JUSQUA $\square$ PREMIER

DE $\square$ PREMIER $\square$ JUSQUA $\square$ DERNIER

DE $\square$ DERNIER $\square$ JUSQUA $\square \langle \text{expqua} \rangle$

DE $\square$ DERNIER $\square$ JUSQUA $\square$ DERNIER

DE $\square$ DERNIER $\square$ JUSQUA $\square$ PREMIER

L'expression $\langle \text{expqua} \rangle$ désigne l'un des index attachés à l'entrée considérée de la file.

L'utilisation d'une des expressions :

$$\text{DE } \square \langle \text{expqua } 1 \rangle \square \text{JUSQUA } \square \left\{ \begin{array}{c} \text{PREMIER} \\ \text{DERNIER} \\ \langle \text{expqua } 2 \rangle \end{array} \right\}$$

$$\text{DE } \square \text{PREMIER } \square \text{JUSQUA } \square \left\{ \begin{array}{c} \langle \text{expqua } 1 \rangle \\ \text{DERNIER} \end{array} \right\}$$

$$\text{DE } \square \text{DERNIER } \square \text{JUSQUA } \square \left\{ \begin{array}{c} \langle \text{expqua } 1 \rangle \\ \text{DERNIER} \end{array} \right\}$$

dans la mesure où la valeur de l'index désigné par $\langle \text{expqua } 1 \rangle$ est différente respectivement de :

- la valeur de l'index désigné par $\langle \text{expqua } 2 \rangle$
- PREMIER
- DERNIER

désigne pour l'entrée de file correspondante, une partie (ou la totalité) des éléments de l'entrée considérée ; en particulier :

- DE $\square$ PREMIER $\square$ JUSQUA $\square$ DERNIER

désigne l'ensemble des éléments de l'entrée correspondante de la file

- DE $\square$ DERNIER $\square$ JUSQUA $\square$ PREMIER

désigne l'ensemble des éléments de l'entrée correspondante de la file réarrangée dans l'ordre inverse, i.e. PREMIER devenant DERNIER et réciproquement.

- DE $\square$ PREMIER $\square$ JUSQUA $\square \langle \text{expqua } 1 \rangle$

désigne l'ensemble des éléments de l'entrée correspondante de la file, depuis le premier jusqu'à celui qui est référencé par la valeur de l'index désigné par $\langle \text{expqua } 1 \rangle$.

- DE □ DERNIER □ JUSQUA □ < expqua 1 >

désigne l'ensemble des éléments de l'entrée correspondante de la file depuis le dernier jusqu'à celui référencé par la valeur de l'index désigné par < expqua 1 >.

- DE □ < expqua 1 > □ JUSQUA □ < expqua 2 >

désigne l'ensemble des éléments de l'entrée correspondante de la file depuis l'élément référencé par la valeur de l'index désigné par < expqua 1 > jusqu'à celui référencé par la valeur de l'index désigné par < expqua 2 >.

Si < expqua 1 > et < expqua 2 > désignent le même index, ou désignent deux index différents mais de valeur identique, un seul élément est concerné

L'expression :

DE □ PREMIER □ JUSQUA □ PREMIER

désigne le premier élément de l'entrée correspondante de la file

L'expression :

DE □ DERNIER □ JUSQUA □ DERNIER

désigne le dernier élément de l'entrée correspondante de la file

Dans le cas où l'expression est de la forme :

DE □ PREMIER □ JUSQUA □ DERNIER

on peut la remplacer par le caractère '.' (point).

De telles expressions qui désignent une partie de file placée en mémoire, sont utilisées notamment dans les appels de procédure.

2.5.3.2.3 - ENSEMBLE D'ELEMENTS D'UNE FILE

Le format général est :

< nom de file > □ LIMITEA □ < expqua >

Cette expression désigne l'ensemble des éléments de file désignés par < expqua > (un identificateur ou une expression qualifiée).

< expqua > désigne un élément de la partition qui structure la file.

REMARQUE :

Dans tous les cas, l'objet désigné par une quelconque expression qualifiée a le même statut (DONNÉE, MIXTE, RESULTAT ou AUXILIAIRE), comme il a été évoqué à plusieurs reprises précédemment, que l'article ou la file dont il est extrait. e.g.

Soient les descriptions :

AUXILIAIRE & FILE & F1 & 20, 30, 30, 20 & PART1 & & 1 & I, J, K &

& & & & & & 2 & L, M, N &

& & & & & & 3 & Ø &

& & & & & & 4 & P, Q & MEMØIRE ;

AUXILIAIRE & ARTICLE & INDF1 & & PART2 & ;

AUXILIAIRE & ARTICLE & MEM & & PART3 & ;

PART1 & & & & ENTIER & 6 & CLEF &

& & & & BØØLEEN & 32 & PHI &

& & & & PARTITION & & PART3 &

& & & & INDEX & 4 & L &

& CLEF DE PART1 = 10 & & & NATUREL & 6 & BETA &

& CLEF DE PART1 = 20 & & & DECIMAL & 15.2 & GAMMA &

& CLEF DE PART1 = 10 & & & MØT & 8 & CØMMØN ;

PART2 & & & & INDEX & 2 & I &

& & & & INDEX & 2 & J &

& & & & INDEX & 2 & K &

& & & & INDEX & 2 & M &

& & & & INDEX & 2 & N &

& & & & INDEX & 2 & Ø &

& & & INDEX & 2 & P &

& & & INDEX & 2 & $\emptyset$;

PART3 & & 10 & REEL & 2,7 & X ;

Les expressions qualifiées :

$P \sqsubseteq DE \sqsubseteq PART2 \sqsubseteq DE \sqsubseteq INDF1$

$M \sqsubseteq DE \sqsubseteq PART2 \sqsubseteq DE \sqsubseteq INDF1$

$X(5) \sqsubseteq DE \sqsubseteq PART3 \sqsubseteq DE \sqsubseteq MEM$

désignent respectivement :

– P , index de la 4ème entrée de la file $F1$, élément de l'article $INDF1$

– M , index de la 2ème entrée de la file $F1$, élément de l'article $INDF1$

– le 5ème élément répété 1ère forme de la suite X de l'article MEM

L'expression qualifiée

$F1(J \sqsubseteq DE \sqsubseteq INDF1, N \sqsubseteq DE \sqsubseteq INDF1, \emptyset \sqsubseteq DE \sqsubseteq INDF1, P \sqsubseteq DE \sqsubseteq INDF1)$

désigne un élément de la file $F1$, structuré par la partition $PART1$, correspondants aux valeurs des index désignés par ;

$J \sqsubseteq DE \sqsubseteq INDF1$

$N \sqsubseteq DE \sqsubseteq INDF1$

$\emptyset \sqsubseteq DE \sqsubseteq INDF1$

$P \sqsubseteq DE \sqsubseteq INDF1$

L'expression qualifiée :

$F1(I \sqsubseteq DE \sqsubseteq INDF1, DE \sqsubseteq L \sqsubseteq DE \sqsubseteq PART1 \sqsubseteq DE \sqsubseteq F1(J \sqsubseteq DE \sqsubseteq INDF1,$

$N \sqsubseteq DE \sqsubseteq INDF1, \emptyset \sqsubseteq DE \sqsubseteq INDF1, P \sqsubseteq DE \sqsubseteq INDF1) \sqsubseteq JUSQUA \sqsubseteq M \sqsubseteq$

$DE \sqsubseteq INDF1, \emptyset \sqsubseteq DE \sqsubseteq INDF1, .)$

désigne une file à deux entrées,

La 1ère entrée va des valeurs d'index

$L \square DE \square PART1 \square DE \square F1 (J \square DE \square INDF1, N \square DE \square INDF1, \emptyset \square DE \square INDF1,$
 $P \square DE \square INDF1)$

jusqu'à $M \square DE \square INDF1$

La 2ème entrée va de PREMIER jusqu'à DERNIER (.) .

L'expression qualifiée :

$F1 \square LIMITEA \square X(2) \square DE \square PART3 \square DE \square PART1$

désigne une file à quatre entrées dont les éléments sont les $X(2) \square DE \square PART3 \square DE \square PART1$.

2.5.4 - Simplification des expressions qualifiées

Comme on y a fait allusion plus haut (2.5.1.6.3) on peut simplifier une expression qualifiée, notamment dans les cas suivants :

l'objet n'est apparu qu'une seule fois dans les descriptions, soit en phrase de type 2 soit dans la description d'une partition, en tant qu'élément simple, qui ne structure qu'un seul article.

L'identificateur suffit alors à désigner l'objet.

- l'une des partitions dans la séquence des noms de partition de l'expression qualifiée n'intervient qu'une seule fois, de façon inconditionnelle et non répétée, dans la description de la partition citée en fin d'expression qualifiée (avant le nom d'article ou de file) ; il est alors possible d'émettre les noms de partition depuis le nom de cette partition jusqu'au nom de l'article ou de la file

Ce ne sont là que deux cas particuliers de la règle générale suivante :

- Il est possible d'omettre dans une expression qualifiée tous les qualificatifs qui n'apportent pas de précision réelle.

REMARQUE :

Les éléments de partition qui n'ont pas reçu de nom ne peuvent évidemment pas être désignés.

2.6 - DEFINITIONS

Les définitions sont exprimées par des phrases de type 5.

Leur format général est :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\} \& \langle \text{valeur} \rangle \left[\left[\{, \langle \text{valeur} \rangle \{ \dots \} \} \mid \langle \text{valeur} \rangle \left[\{, \langle \text{valeur} \rangle \{ \dots \} \right] \right] \right] ;$$

Une définition fixe la valeur d'un objet dans le contexte d'un programme (i.e. un traitement immédiat ou une procédure) ; ce programme n'y peut rien changer par la suite

Un objet ainsi défini devient en quelque sorte une constante et l'expression qualifiée ou l'identificateur qui le désigne joue, dans le programme, un rôle analogue à celui de 40 ou de 'A', par exemple.

Les objets qui font l'objet d'une définition sont de statut AUXILIAIRE et de support MEMØIRE obligatoirement. Ils doivent satisfaire en outre à certaines conditions. Pour les énoncer plus facilement, on va classer les objets de statut AUXILIAIRE et de support MEMØIRE en trois catégories :

- les objets qui peuvent être définis,
- les objets qui doivent être définis,
- les objets qui ne peuvent pas être définis

Les premières catégories vont être examinées successivement, la troisième catégorie se déduit des deux premières par élimination.

2.6.1 - Objets pouvant être définis

Peuvent être définis :

- les objets simples fondamentaux, i.e. les objets de l'une des natures suivantes :

MØT

BØØLEEN

NATUREL

ENTIER

DECIMAL

REEL

dont la longueur est fixée par une constante (ou par un objet défini) et dont les facteurs de répétition éventuels, sont constants (i.e. explicités dans la description ou faisant l'objet d'une définition). Ces dernières restrictions concernant :

< nb fois >

< drapeau 1 > et < longueur maxi > ;

- les objets de nature INDEX ;
- les objets de nature PRØCEDURE ;
- les objets de nature ETIQUETTE ;
- les articles, non structurés par une sous-partition (2.4.5), dont les éléments sont des objets simples fondamentaux, des index, des procédures, des étiquettes ;
- les files, non structurées par une sous-partition (2.4.5), dont chaque entrée a une longueur fixée par une constante et dont les articles satisfont aux conditions ci-dessus ;
- un objet de nature ØRDRE

2.6.2 - Objets devant être définis

Doivent être définis, s'ils sont de statut AUXILIAIRE :

- les objets de nature PRØCEDURE ;
- les objets de nature ETIQUETTE ;
- < drapeau 1 >, < longueur maxi > et/ou < nb fois > pour les éléments répétés.:

Les objets de nature ETIQUETTE ne peuvent être que de statut AUXILIAIRE. Par contre, les objets de nature PRØCEDURE et les paramètres des facteurs de répétition peuvent être de statut DØNNEE ; dans ce cas il est interdit de les définir, puisque les objets de statut DØNNEE ont une valeur fixée hors de la procédure.

Une grande liberté est laissée en ce qui concerne les variables de nature ~~ORDRE~~, il est possible de ne pas les définir et donc d'opérer des modifications, durant le traitement, sur cet ordre.

2.6.3 - Introduction des définitions dans les programmes

Une définition est introduite dans un programme (i.e. un traitement immédiat ou une procédure) par une phrase de type 5 en plaçant :

- dans la zone "nom", l'expression qualifiée ou l'identificateur désignant l'objet à définir ;
- dans la zone "valeur", la valeur de cet objet, valeur dont l'expression obéit aux règles ci-dessous ;
- un point virgule qui marque la fin de la définition.

Règles d'écriture des valeurs :

1 - ~~MOT~~

La valeur d'un tel objet est désignée par une constante ou une suite de constantes de mot :

' < chaîne de caractères > "

< chaîne de caractères > ne doit pas contenir les caractères ' et ", ceux-ci étant cités par les mots réservés G (G et G (G respectivement.

Le nombre de caractères de < chaîne de caractères > est limité à 256 (G (G et G) G comptent pour un seul caractère évidemment).

Les constantes de mot sont séparées par des virgules dans le cas où l'objet à définir est affecté d'un facteur de répétition. e.g.

Soit la description de la partition LAMBDA structurant l'article ART02 de statut AUXILIAIRE et de support ~~MEMOIRE~~ :

```
LAMBDA  & & & & MOT & 18 & ELEMENT 1 &
        & & & 4 & MOT & 3 & ELEMENT 2 &
        & & ' * * ', 6 & & MOT & 2 & ELEMENT 3 &
        & & ' / / ', 5 & 3 & MOT & 2 & ELEMENT 4 ;
```

Les définitions éventuelles des éléments de ART02 sont de la forme :

ELEMENT 1 □ DE □ LAMBDA □ DE □ ART02 & 'G(GENTRE □ GUILLEMETS G)G' ;

ELEMENT 2 □ DE □ LAMBDA □ DE □ ART02 & 'ABC", 'DEF", 'GHI" ;

ELEMENT 3 □ DE □ LAMBDA □ DE □ ART02 & 'X1", '+ 2", '- ", 'X3", '***" ;

ELEMENT 4 □ DE □ LAMBDA □ DE □ ART02 & 'ØP", 'QR", "ST", 'UV", 'XY", 'Z1", '32",
'56", '7B", 'DF", '77" ;

Les valeurs des éléments sont :

ELEMENT 1 □ DE □ LAMBDA □ DE □ ART02 — ' ENTRE □ GUILLEMETS"

ELEMENT 2 (1) _____ ABC

ELEMENT 2 (2) _____ DEF

ELEMENT 2 (3) _____ GHI

ELEMENT 2 (4) n'a pas reçu de valeur et n'est donc pas défini, il est considéré comme un objet de statut AUXILIAIRE.

ELEMENT 3 (1) □ DE □ LAMBDA □ DE □ ART02 X1

ELEMENT 3 (2) _____ + 2

ELEMENT 3 (3) _____ -

ELEMENT 3 (4) _____ X3

ELEMENT 3 (4) est le dernier élément de la répétition puisque la valeur de ELEMENT 3 (5) est celle du drapeau.

ELEMENT 4 (1) (1) □ DE □ LAMBDA □ DE □ ART02 ØP

ELEMENT 4 (2) (1) _____ QR

ELEMENT 4 (3) (1) _____ ST

ELEMENT 4 (1) (2) _____ UV

ELEMENT 4 (2) (2) _____ XY

ELEMENT 4 (3) (2)	_____	Z1
ELEMENT 4 (1) (3)	_____	32
ELEMENT 4 (2) (3)	_____	56
ELEMENT 4 (3) (3)	_____	7B
ELEMENT 4 (1) (4)	_____	DF

ELEMENT 4 (1) (4) est le dernier élément de la répétition puisque la valeur de ELEMENT 4 (2) (4) est celle du drapeau.

Il ne sera pas possible, dans le traitement, d'attribuer une valeur aux éléments restants puisque la position du drapeau '//' est définie.

2 - ~~BOOLEEN~~

La valeur d'un tel objet est désignée par une constante de nombre booléen ou, si l'objet est répété, une suite de constantes de nombre booléen séparées par des virgules ; chaque constante doit être de la forme :

$$\left\{ \begin{array}{c} \text{VRAI} \\ \text{FAUX} \end{array} \right\} \left[\left\{ \right\} \square \left\{ \begin{array}{c} \text{VRAI} \\ \text{FAUX} \end{array} \right\} \dots \right]$$

ou encore :

$$\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} \left[\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} \dots \right]$$

VRAI et 1, FAUX et 0 ayant, dans ce cas la même signification respectivement.

3 - NATUREL

La valeur d'un tel objet est désignée par une constante ou une suite de constantes de nombre naturel séparées par des virgules dans le cas où l'objet est répété, la constante étant un nombre sans signe

4 - ENTIER

La valeur d'un tel objet est désignée par une constante ou une suite de constantes de nombre entier séparées des virgules, la constante étant de la forme :

$$\left[\left\{ \begin{array}{c} + \\ - \end{array} \right\} \right] < \text{constante de nombre naturel} >$$

5 - DECIMAL

La valeur d'un tel objet est désignée par une constante ou une suite de constantes de nombre décimal séparées par des virgules, la constante étant de la forme :

$$\left[\left\{ \begin{array}{c} + \\ - \end{array} \right\} \right] \cdot \text{constante de nombre naturel} \cdot \left[\left[< \text{constante de nombre naturel} > \right] \right]$$

6 - REEL

La valeur d'un tel objet est désignée par une constante ou une suite de constantes de nombre réel séparées par des virgules, la constante étant de la forme :

$$\left[\left\{ \begin{array}{c} + \\ - \end{array} \right\} \right] < \text{constante de nombre naturel} > \left[\left[< \text{constante de nombre naturel} > \right] \right]$$

$$\left[E \left\{ \begin{array}{c} + \\ 0 \\ - \end{array} \right\} < \text{constante de nombre naturel} > \right]$$

7 - INDEX

La valeur d'un tel objet est désignée par un pseudo-terme d'index (2.7.7) ou une suite de pseudos-termes d'index séparés par des virgules.

8 - ARTICLE

La valeur d'un tel objet est désignée par une suite composée des constantes qui désignent les valeurs de chacun des objets qui le composent (d'après la décomposition décrite par une phrase de type 3 ou 4) séparées par des virgules.

9 - FILE

La valeur d'un tel objet est désignée par une suite de définitions d'article séparées par des barres ' | '

Pour une file à une entrée, on définit ainsi, successivement, toutes les valeurs des articles successifs, dans l'ordre croissant des valeurs d'index (i.e. de PREMIER vers DERNIER)

Pour une file à plusieurs entrées, tout se passe comme si on la ramenait à une file à une entrée en faisant varier d'autant plus rapidement un index attaché à une entrée que le numéro de celle-ci est plus élevé.

10 - PROCEDURE

La valeur d'un tel objet est désignée par une suite d'identificateurs séparés par des virgules, ces identificateurs étant des noms de procédure.

Il est rappelé que seuls les huit premiers caractères d'un nom de procédure sont significatifs, les caractères restants sont ignorés.

11 - ØRDRE

La valeur d'un tel objet est désignée par une suite de caractères cités - c'est-à-dire placés entre un guillemet ouvrant ' et un guillemet fermant " - séparés par des virgules.

Cette définition doit respecter les règles formulées au paragraphe 2.3.5.3.

Il est rappelé qu'un seul objet de nature ØRDRE peut être décrit dans un programme.

12 - ETIQUETTE

La valeur d'un tel objet est désigné par une suite d'identificateurs séparés par des virgules, ces identificateurs étant des étiquettes de la procédure.

REMARQUE :

Comme dans les instructions d'attribution (2 8 1) les valeurs des mots sont cadrées à gauche, celles des nombres booléens, naturels, entiers sont cadrées à droite, celles des nombres décimaux sont cadrées sur la place de la virgule, dans les emplacements définis par leurs longueurs. Les nombres réels (virgule flottante) sont normalisés dans les emplacements définis par leurs longueurs.

Dans tous les autres cas, si la valeur de définition de l'objet n'est pas exprimable dans l'emplacement qui lui est réservé, la définition est erronée et le programme incorrect.

Ces emplacements sont complétés à droite par des blancs () pour les mots, par une suite de valeurs FAUX (ou 0) à gauche par les booléens, par des zéros en tête pour les nombres naturels, entiers ou décimaux ; pour ces derniers, la partie décimale de la valeur est tronquée ou complétée à droite par des zéros suivant qu'elle occupe plus ou moins de place qu'il n'est prévu pour son expression. L'exposant et la mantisse d'un nombre réel sont cadrés et complétés comme s'il s'agissait respectivement d'un entier et de la partie fractionnaire d'un nombre décimal.

Un même objet ne peut être défini qu'une seule fois, autrement dit :

- Un même identificateur ou une même expression qualifiée ne peut figurer qu'une seule fois dans la zone "nom" des phrases de type 5 d'un programme.

2.7 - TERMES

Comme on l'a dit plus haut un traitement est composé :

- d'un traitement immédiat ;
- éventuellement d'une ou de plusieurs procédures

On reviendra plus loin - quand on abordera la partie traitement - sur les caractères spécifiques de ces deux espèces de programme.

Auparavant, il importe de définir les termes.

2.7.1 - Usage des termes - Forme générale

Les termes sont utilisés pour désigner les objets obtenus par des opérations portant sur d'autres objets plus simples ; les termes doivent donc mentionner ces derniers objets et les opérations effectuées.

Il existe plusieurs formes de termes :

1 - Arithmétique à deux facteurs

$$\left(\left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\} \right) < \text{opération} > \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\}$$

2 - Arithmétique à un facteur et "arrondi à" éventuel

$$\left(\left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \right) \left[\text{ARRONDIA} \right] \left\{ \begin{array}{l} \text{INFER} \\ \text{PROCHE} \\ \text{SUPER} \end{array} \right\}$$

3 - Arithmétique à deux facteurs et "arrondi à"

$$\left(\left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \right) < \text{opération} > \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \left[\text{ARRONDIA} \right] < \text{terme arrondi} >$$

$$\left\{ \begin{array}{l} \text{INFER} \\ \text{PROCHE} \\ \text{SUPER} \end{array} \right\}$$

4 - Arithmétique à trois facteurs et "reste"

$$\left\{ \left[\begin{array}{l} + \\ - \end{array} \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} / \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \square \text{ RESTE } \square \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\} \right\}$$

5 - Conversion

$$\left\{ \begin{array}{l} \text{NATUREL} \\ \text{ENTIER} \\ \text{DECIMAL} \\ \text{REEL} \\ \text{INDEX} \\ \text{BØØLEEN} \end{array} \right\} \square \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\}$$

6 - Conversion - MØT

$$\text{MØT} \square \left\{ \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\} \left[\square \left\{ \begin{array}{l} \text{INFER} \\ \text{PRØCHE} \\ \text{SUPER} \end{array} \right\} \right] \right\} \\ < \text{constante} >$$

7 - Conversion arithmétique

7.1 - Opération ABS

$$\left[\left\{ \begin{array}{l} + \\ - \end{array} \right\} \right] \text{ ABS } \square \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\}$$

7.2 - Opération SIGNE

$$\text{SIGNE} \quad \square \quad \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\}$$

7.3 - Opération MOD et "arrondi à" éventuel

$$\left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \quad \square \quad \text{MOD} \quad \square \quad \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \left[\begin{array}{c} \square \text{ARRONDIA} \square \\ \\ \end{array} \right]$$

$$< \text{terme d'arrondi} > \quad \square \quad \left\{ \begin{array}{l} \text{INFER} \\ \text{PRØCHE} \\ \text{SUPER} \end{array} \right\} \quad \left. \vphantom{\left\{ \begin{array}{l} \text{INFER} \\ \text{PRØCHE} \\ \text{SUPER} \end{array} \right\}} \right]$$

8 - BØØLEEN

8.1 - Opération NØN

$$\text{NØN} \quad \square \quad \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \left[\begin{array}{c} \square \left\{ \begin{array}{l} \text{ØU} \\ \text{ET} \end{array} \right\} \square \left[\text{NØN} \square \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \end{array} \right]$$

8.2 - Opération ØU

$$\left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \quad \square \quad \text{ØU} \quad \square \quad \left[\text{NØN} \square \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\}$$

8.3 - Opération ET

$$\left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\} \quad \square \quad \text{ET} \quad \square \quad \left[\text{NØN} \square \right] \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \\ < \text{constante} > \end{array} \right\}$$

9 - Condition

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\} \left[\begin{array}{l} \left\{ \begin{array}{l} = \\ < \\ < = \\ > \\ > = \\ \neq \end{array} \right\} \\ \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\} \end{array} \right]$$

10 - Longueur

10.1 - Arithmétique

$$\left\{ \begin{array}{l} \emptyset \text{RD} \\ \text{PRE} \end{array} \right\} \square \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

10.2 - File

$$\text{LONG} \square \left\{ \begin{array}{l} \langle \text{expression de file} \rangle \\ \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

11 - INDEX

11.1 - Double-relatif

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\} \square \left\{ \begin{array}{l} \text{AVANT} \\ \text{APRES} \end{array} \right\} \square \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

11.2 - Simple-relatif

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\} \square \left\{ \begin{array}{l} \text{APRES} \square \text{PREMIER} \\ \text{AVANT} \square \text{DERNIER} \end{array} \right\}$$

11.3 - Simple

$$\left\{ \begin{array}{l} \text{PREMIER} \\ \text{DERNIER} \end{array} \right\}$$

Les termes servent à désigner des objets de l'une ou l'autre des natures suivantes :

NATUREL

ENTIER

DECIMAL

REEL

~~MOT~~

INDEX

~~BOOLEEN~~

Les termes de longueur et de condition désignent respectivement des objets de nature :

NATUREL

et ~~BOOLEEN~~ (VRAI ou FAUX)

On va maintenant examiner les termes en fonction de la nature des objets qu'ils désignent.

2.7.2 - Termes désignant des nombres naturels

Ils sont de l'une ou l'autre des formes :

1. Arithmétique à deux facteurs
2. Arithmétique à un facteur et "arrondi à" éventuel
3. Arithmétique à deux facteurs et "arrondi à"
5. Conversion

7.3 ~~MOT~~ et "arrondi à" éventuel

Les facteurs sont les suivants :

1.

$$\text{Si } \left\{ \begin{array}{l} \langle \text{expqua } 1 \rangle \\ \langle \text{identificateur } 1 \rangle \\ \langle \text{constante } 1 \rangle \end{array} \right\} \quad \text{et} \quad \left\{ \begin{array}{l} \langle \text{expqua } 2 \rangle \\ \langle \text{identificateur } 2 \rangle \\ \langle \text{constante } 2 \rangle \end{array} \right\} \quad \text{désignent}$$

l'un et l'autre des nombres naturels ; soit $\langle \text{terme } 1 \rangle$ et $\langle \text{terme } 2 \rangle$ les deux facteurs ci-dessus respectivement, alors :

$$- \langle \text{terme } 1 \rangle + \langle \text{terme } 2 \rangle$$

désigne la somme de $\langle \text{terme } 1 \rangle$ et $\langle \text{terme } 2 \rangle$:

$$- \langle \text{terme } 1 \rangle - \langle \text{terme } 2 \rangle$$

désigne la différence de $\langle \text{terme } 1 \rangle$ et $\langle \text{terme } 2 \rangle$:

Si $\langle \text{terme } 1 \rangle$ est strictement inférieur à $\langle \text{terme } 2 \rangle$ ce terme n'a pas de signification et sa présence dans un programme en arrête l'exécution.

$$- \langle \text{terme } 1 \rangle * \langle \text{terme } 2 \rangle$$

désigne le produit de $\langle \text{terme } 1 \rangle$ par $\langle \text{terme } 2 \rangle$

$$- \langle \text{terme } 1 \rangle / \langle \text{terme } 2 \rangle$$

désigne le quotient de $\langle \text{terme } 1 \rangle$ par $\langle \text{terme } 2 \rangle$ à une unité près par défaut.

Si $\langle \text{terme } 2 \rangle$ désigne 0, le terme $\langle \text{terme } 1 \rangle / \langle \text{terme } 2 \rangle$ n'a pas de signification et sa présence dans un programme en arrête l'exécution.

$$- \langle \text{terme } 1 \rangle \text{ MOD } \langle \text{terme } 2 \rangle$$

désigne le reste de la division de $\langle \text{terme } 1 \rangle$ par $\langle \text{terme } 2 \rangle$; même remarque que précédemment si $\langle \text{terme } 2 \rangle$ désigne zéro.

2. Peu importe, ici, que l'expression qui précède $\square \text{ ARRONDIA } \square \dots$ soit composée d'un ou de deux facteurs ; soit donc $\langle \text{terme } 3 \rangle$ défini par :

$$\left\{ \begin{array}{l} \left\{ \begin{array}{l} \langle \text{expqua } 1 \rangle \\ \langle \text{identificateur } 1 \rangle \\ \langle \text{constante } \rangle \end{array} \right\} \\ \langle \text{l'un des termes précédents} \rangle \end{array} \right\}$$

et désignant un nombre naturel.

Le format général devient ainsi :

$$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square \left\{ \begin{array}{c} 10 \text{ AINE} \\ 100 \text{ AINE} \\ 1000 \text{ AINE} \\ \dots \end{array} \right\} \square \left\{ \begin{array}{c} \text{INFER} \\ \text{PRØCHE} \\ \text{SUPER} \end{array} \right\}$$

ou encore :

$$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square 10 \left[\{ 0 \} \dots \right] \text{ AINE} \square \left\{ \begin{array}{c} \text{INFER} \\ \text{PRØCHE} \\ \text{SUPER} \end{array} \right\}$$

Ce terme tout entier désigne un nombre naturel, à savoir : le nombre naturel désigné par $\langle \text{terme}_3 \rangle$ arrondi sur la position décimale spécifiée par le facteur d'arrondi et la borne relative (INFER, PRØCHE ou SUPER)

Ainsi :

$$328 \square \text{ARRONDIA} \square 10 \text{ AINE} \square \text{INFER}$$

$$328 \square \text{ARRONDIA} \square 10 \text{ AINE} \square \text{PRØCHE}$$

$$328 \square \text{ARRONDIA} \square 10 \text{ AINE} \square \text{SUPER}$$

désignent, respectivement, les nombres naturels :

$$320, 330, 330$$

REMARQUE :

Par convention

$$325 \text{ ARRONDIA } 10 \text{ AINE PRØCHE}$$

par exemple, désigne 330, c'est-à-dire le nombre :

$$\langle \text{terme}_3 \rangle + (10/2)$$

arrondi à la dizaine inférieure.

3.

Soit $\langle \text{terme}_3 \rangle$ ainsi défini :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

Si $\langle \text{terme}_3 \rangle$ désigne un nombre entier positif ou nul, un nombre décimal positif ou nul de la forme (N.O), un nombre booléen ou un mot composé uniquement de chiffres, alors :

$$\text{NATUREL} \sqsubset \langle \text{terme}_3 \rangle$$

est un terme qui désigne un nombre naturel (voir 2.8.3).

La signification des termes de longueur qui désignent des nombres naturels est expliquée au paragraphe 2.7.10.

2.7.3 - Termes désignant des nombres entiers

Ils sont de l'une ou l'autre des formes :

1. Arithmétique à deux facteurs
2. Arithmétique à un facteur et "arrondi à" éventuel
3. Arithmétique à deux facteurs et "arrondi à"
5. Conversion

7.3 MOD et "arrondi à" éventuel

Les facteurs sont les suivants :

1.

$$\text{Si } \left\{ \left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{l} \langle \text{expqua}_1 \rangle \\ \langle \text{identificateur}_1 \rangle \\ \langle \text{constante}_1 \rangle \end{array} \right\} \right\} \text{ et } \left\{ \begin{array}{l} \langle \text{expqua}_2 \rangle \\ \langle \text{identificateur}_2 \rangle \\ \langle \text{constante}_2 \rangle \end{array} \right\} \text{ désignent}$$

l'un et l'autre des nombres entiers; soit $\langle \text{terme}_1 \rangle$ et $\langle \text{terme}_2 \rangle$ les deux facteurs ci-dessus, respectivement ; $\langle \text{constante}_2 \rangle$ étant une constante sans signe, alors :

$$- \langle \text{terme}_1 \rangle + \langle \text{terme}_2 \rangle$$

désigne la somme de $\langle \text{terme}_1 \rangle$ et $\langle \text{terme}_2 \rangle$

$$- \langle \text{terme}_1 \rangle - \langle \text{terme}_2 \rangle$$

désigne la différence de $\langle \text{terme}_1 \rangle$ et $\langle \text{terme}_2 \rangle$

$$- \langle \text{terme}_1 \rangle * \langle \text{terme}_2 \rangle$$

désigne le produit de $\langle \text{terme}_1 \rangle$ et de $\langle \text{terme}_2 \rangle$, le signe du résultat étant le produit des signes.

$$- \langle \text{terme}_1 \rangle / \langle \text{terme}_2 \rangle$$

désigne le quotient de $\langle \text{terme}_1 \rangle$ par $\langle \text{terme}_2 \rangle$ ainsi défini : la valeur absolue du quotient est le quotient à une unité près par défaut des valeurs absolues des deux facteurs ; son signe est le produit des signes des deux facteurs. Si $\langle \text{terme}_2 \rangle$ désigne 0, même remarque que pour le quotient de nombres naturels

$$- \langle \text{terme}_1 \rangle \square \text{MOD} \square \langle \text{terme}_2 \rangle$$

désigne le reste de la division de $\langle \text{terme}_1 \rangle$ par $\langle \text{terme}_2 \rangle$, i.e.

$$\langle \text{terme}_1 \rangle - (\langle \text{terme}_1 \rangle / \langle \text{terme}_2 \rangle) * \langle \text{terme}_2 \rangle$$

si $\langle \text{terme}_2 \rangle$ désigne 0, même remarque que pour les nombres naturels.

2. Peu importe, ici, que le terme qui précède $\square \text{ARRONDIA} \square \dots$ soit composé d'un ou de deux facteurs ; soit donc $\langle \text{terme}_3 \rangle$, désignant un nombre entier, défini par :

$$\left(\begin{array}{l} \left[\begin{array}{l} + \\ - \end{array} \right] \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\} \\ \langle \text{l'un des termes précédents} \rangle \end{array} \right)$$

Le format général devient ainsi :

$$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square 10 \left[\begin{array}{l} \{0\} \dots \end{array} \right] \text{AINE} \square \left\{ \begin{array}{l} \text{INFER} \\ \text{PROCHE} \\ \text{SUPER} \end{array} \right\}$$

Ce terme désigne un nombre entier, qui est le nombre entier désigné par $\langle \text{terme}_3 \rangle$ arrondi sur la position décimale spécifiée par le facteur d'arrondi et la borne relative.

Ainsi les termes :

$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square 10 \text{ AINE} \square \text{INFER}$

$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square 10 \text{ AINE} \square \text{PROCHE}$

$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square 10 \text{ AINE} \square \text{SUPER}$

désignent respectivement les nombres :

$+ 570, + 580, + 580$

si $\langle \text{terme}_3 \rangle$ désigne $+ 575$

ou, si $\langle \text{terme}_3 \rangle$ désigne $- 882$,

$- 880, - 880, - 890$

i.e., la borne relative spécifiée par l'un des mots réservés INFER, PROCHE ou SUPER concerne la valeur absolue.

3. Avec $\langle \text{terme}_2 \rangle$ défini en 1 ci-dessus :

- $+ \langle \text{terme}_2 \rangle$

désigne : plus $\langle \text{terme}_2 \rangle$ (Si $\langle \text{terme}_2 \rangle$ est une constante, celle-ci n'a pas de signe), ainsi, lorsque $\langle \text{terme}_2 \rangle$ est 024, $+ \langle \text{terme}_2 \rangle$ désigne le nombre entier $+ 24$.

- $- \langle \text{terme}_2 \rangle$

désigne : $- \langle \text{terme}_2 \rangle$

- $[+] \text{ ABS} \square \langle \text{terme}_2 \rangle$

désigne la valeur absolue de $\langle \text{terme}_2 \rangle$

- $- \text{ABS} \square \langle \text{terme}_2 \rangle$

désigne moins (valeur absolue de $\langle \text{terme}_2 \rangle$)

- $\text{SIGNE} \square \langle \text{terme}_2 \rangle$

désigne le nombre entier $- 1, 0$ ou $+ 1$ selon que $\langle \text{terme}_2 \rangle$ est < 0 , $= 0$ ou > 0 .

4. Conversion

Soit $\langle \text{terme}_4 \rangle$ défini par

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

désignant un nombre naturel, décimal ou réel ou un mot composé de chiffres éventuellement précédés du signe + ou du signe -, alors :

$$\text{ENTIER} \sqsupset \langle \text{terme}_4 \rangle$$

désigne un nombre entier. e.g. :

$$\text{ENTIER} \sqsupset \langle \text{terme}_4 \rangle$$

désigne + 128

si $\langle \text{terme}_4 \rangle$ désigne :

- le nombre naturel 128
- le nombre décimal 128.2
- le nombre réel + 0.128982 E + 03
- le mot + 128

Ainsi :

$$\text{ENTIER} \sqsupset \left\{ \begin{array}{l} - 3.74 \\ 128.5 \end{array} \right\}$$

désigne respectivement,

- - 3, i.e. moins (partie entière de la valeur absolue de - 3.74)
- 128, i.e. partie entière de 128.5

2.7.4 - Termes désignant des nombres décimaux

Ils sont de l'une ou l'autre des formes :

1. Arithmétique à deux facteurs
2. Arithmétique à un facteur et "arrondi à" éventuel
3. Arithmétique à deux facteurs et "arrondi à"
4. Arithmétique à trois facteurs "reste"
5. Conversion

Les facteurs sont les suivants :

1.

$$\text{Si } \left\{ \left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{c} < \text{expqua1} > \\ < \text{identificateur 1} > \\ < \text{constante 1} > \end{array} \right\} \right\} \text{ et } \left\{ \begin{array}{c} < \text{expqua2} > \\ < \text{identificateur 2} > \\ < \text{constante 2} > \end{array} \right\} \text{ désignent}$$

l'un et l'autre des nombres décimaux, soit $< \text{terme 1} >$ et $< \text{terme 2} >$ les facteurs ci-dessus respectivement, $< \text{constante 2} >$ étant une constante sans signe, alors :

$$- < \text{terme 1} > + < \text{terme 2} >$$

désigne la somme de $< \text{terme 1} >$ et $< \text{terme 2} >$

$$- < \text{terme 1} > - < \text{terme 2} >$$

désigne la différence de $< \text{terme 1} >$ et $< \text{terme 2} >$

$$- < \text{terme 1} > * < \text{terme 2} >$$

désigne le produit de $< \text{terme 1} >$ et $< \text{terme 2} >$ le signe du résultat étant le produit des signes.

$$- < \text{terme 1} > / < \text{terme 2} >$$

désigne le quotient de $< \text{terme 1} >$ et $< \text{terme 2} >$ tronqué sur la dernière position de la zone du résultat, le signe étant le produit des signes. Si $< \text{terme 2} >$ désigne 0, le terme $< \text{terme 1} > / < \text{terme 2} >$ n'a pas de signification et sa présence dans un programme en arrête l'exécution.

Le reste de la division peut être obtenu.

2. Peu importe, ici, que le terme qui précède $\square$ ~~ARR~~ONDIA $\square$... soit composé d'un ou de deux facteurs ; soit donc $\langle \text{terme}_3 \rangle$, désignant un nombre décimal, défini par :

$$\left\{ \left[\begin{array}{c} + \\ - \end{array} \right] \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{l'un des termes précédents} \rangle \end{array} \right\} \right\}$$

le format général devient ainsi :

$$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square \left\{ \begin{array}{l} 10 \left[\begin{array}{c} 0 \\ \dots \end{array} \right] \left\{ \begin{array}{l} \text{AINE} \\ \text{IEME} \end{array} \right\} \\ \text{UNITE} \end{array} \right\} \square \left\{ \begin{array}{l} \text{INFER} \\ \text{PROCHE} \\ \text{SUPER} \end{array} \right\}$$

Ce terme tout entier désigne un nombre décimal, à savoir : le nombre décimal désigné par $\langle \text{terme}_3 \rangle$ arrondi sur la position décimale spécifiée par le facteur arrondi et la borne relative.

Ainsi les termes :

$$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square 100 \text{ IEME} \square \text{INFER}$$

$$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square 1000 \text{ AINE} \square \text{SUPER}$$

$$\langle \text{terme}_3 \rangle \square \text{ARRONDIA} \square \text{UNITE} \square \text{PROCHE}$$

désignent respectivement :

3127.26

4000.00

3127.00

si $\langle \text{terme}_3 \rangle$ désigne le nombre décimal 3127.262.

Comme précédemment ~~INFER~~, ~~PROCHE~~ et ~~SUPER~~ concernent la valeur absolue.

3. On peut profiter du calcul du quotient de deux nombres décimaux pour obtenir le reste de la division ; on emploie alors un terme de la forme suivante :

$$\langle \text{terme}_1 \rangle / \langle \text{terme}_2 \rangle \square \text{RESTE} \square \langle \text{terme}_4 \rangle$$

avec $\langle \text{terme}_4 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

et désignant un nombre décimal.

L'objet désigné par $\langle \text{terme}_4 \rangle$ reçoit le reste de la division de $\langle \text{terme}_1 \rangle$ par $\langle \text{terme}_2 \rangle$, son signe étant celui de l'objet désigné par $\langle \text{terme}_1 \rangle$.

4. Avec $\langle \text{terme}_2 \rangle$ défini en 1 ci-dessus :

- $+$ $\langle \text{terme}_2 \rangle$

désigne:plus $\langle \text{terme}_2 \rangle$

- $-$ $\langle \text{terme}_2 \rangle$

désigne:moins $\langle \text{terme}_2 \rangle$

- $[+]$ ABS $\square$ $\langle \text{terme}_2 \rangle$

désigne la valeur absolue de $\langle \text{terme}_2 \rangle$

- $-$ ABS $\square$ $\langle \text{terme}_2 \rangle$

désigne:moins (valeur absolue de $\langle \text{terme}_2 \rangle$)

- SIGNE $\square$ $\langle \text{terme}_2 \rangle$

désigne le nombre entier -1 , 0 ou $+1$ selon que le nombre décimal désigné par $\langle \text{terme}_2 \rangle$ est < 0 , $= 0$ ou > 0 .

5. Soit $\langle \text{terme}_5 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

et désignant un nombre naturel, un nombre entier, un nombre réel ou un mot susceptible d'être interprété comme une constante de nombre décimal, alors :

$$\text{DECIMAL} \square \langle \text{terme}_5 \rangle$$

est un terme qui désigne un nombre décimal.

Ainsi :

DECIMAL □ < terme 5 > désigne + 8.0 si < terme 5 > désigne le nombre naturel 8

_____ - 17.0 _____ entier - 17

_____ + 3.14 _____ réel +0.314 E+01

_____ - 17.0 _____ mot - 17

2.7.5 - Termes désignant des nombres réels

Ils sont de l'une des formes :

1. Arithmétique à deux facteurs
2. Arithmétique à un facteur sans "arrondi à"
5. Conversion

1.

Si $\left\{ \begin{bmatrix} + \\ - \end{bmatrix} \right\} \left\{ \begin{matrix} < \text{expqua}_1 > \\ < \text{identificateur}_1 > \\ < \text{constante}_1 > \end{matrix} \right\} \text{ et } \left\{ \begin{matrix} < \text{expqua}_2 > \\ < \text{identificateur}_2 > \\ < \text{constante}_2 > \end{matrix} \right\}$ désignent

l'un et l'autre des nombres réels, soit < terme 1 > et < terme 2 > les deux facteurs ci-dessus respectivement, < constante 2 > étant une constante sans signe, alors :

- < terme 1 > + < terme 2 >

désigne la somme de < terme 1 > et < terme 2 >

- < terme 1 > - < terme 2 >

désigne la différence de < terme 1 > et < terme 2 >

- < terme 1 > * < terme 2 >

désigne le produit de < terme 1 > et < terme 2 >, le signe du résultat étant le produit des signes.

- $\langle \text{terme } 1 \rangle / \langle \text{terme } 2 \rangle$

désigne le quotient de $\langle \text{terme } 1 \rangle$ et $\langle \text{terme } 2 \rangle$, le signe du résultat étant le produit des signes.

Si $\langle \text{terme } 2 \rangle$ désigne un nombre réel nul le terme $\langle \text{terme } 1 \rangle / \langle \text{terme } 2 \rangle$ n'a pas de signification et sa présence dans un programme en arrête l'exécution.

Soit $\langle \text{terme } 3 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

désignant un nombre naturel, entier, décimal ou réel, $\langle \text{constante} \rangle$ étant une constante sans signe ; alors :

$$\langle \text{terme } 1 \rangle ** \langle \text{terme } 3 \rangle$$

désigne le nombre réel, $\langle \text{terme } 1 \rangle$ "élevé à la puissance" $\langle \text{terme } 3 \rangle$.

Dans le cas, où le nombre désigné par $\langle \text{terme } 3 \rangle$ est compris dans l'un deux intervalles :

$$\begin{array}{l}] -1, 0 [\\ \text{ou} \quad] 0, +1 [\end{array}$$

il est nécessaire que le nombre réel désigné par $\langle \text{terme } 1 \rangle$ soit positif ou nul.

Dans le cas où le nombre désigné par $\langle \text{terme } 3 \rangle$ est négatif il est nécessaire que le nombre réel désigné par $\langle \text{terme } 1 \rangle$ soit différent de 0.

Si l'une des deux conditions précédentes (ou les deux) n'est pas satisfaite, le terme n'a pas de signification et sa présence dans un programme en arrête l'exécution.

2. Avec $\langle \text{terme } 2 \rangle$ défini ci-dessus

- $+ \langle \text{terme } 2 \rangle$

désigne:plus $\langle \text{terme } 2 \rangle$

- $- \langle \text{terme } 2 \rangle$

désigne:moins $\langle \text{terme } 2 \rangle$

- $[+]$ ABS $\square$ $\langle \text{terme}_2 \rangle$

désigne la valeur absolue de $\langle \text{terme}_2 \rangle$

- $-$ ABS $\square$ $\langle \text{terme}_2 \rangle$

désigne moins (valeur absolue de $\langle \text{terme}_2 \rangle$)

- SIGNE $\square$ $\langle \text{terme}_2 \rangle$

désigne le nombre entier $-1, 0$ ou $+1$ selon que le nombre réel désigné par $\langle \text{terme}_2 \rangle$ est $< 0, = 0, > 0$.

3. Soit $\langle \text{terme}_4 \rangle$ ainsi défini :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

désignant un nombre naturel, un nombre entier, un nombre décimal ou un mot susceptible d'être interprété comme une constante de réel, alors :

$$\text{REEL} \square \langle \text{terme}_4 \rangle$$

est un terme qui désigne un nombre réel, ainsi :

REEL $\square$ $\langle \text{terme}_4 \rangle$ désigne $+0.8 \text{ E } +01$ si $\langle \text{terme}_4 \rangle$ désigne le nombre naturel 8

_____ $-0.17 \text{ E } +02$ _____ entier -17

_____ $+0.12805 +03$ _____ décimal 128.05

_____ $+0.153624 \text{ E } +06$ _____ mot 153624.0

$\langle \text{REEL} \rangle \square 'Z'$ n'a pas de signification.

2.7.6 - Termes désignant des mots

Ils sont de l'une ou l'autre des formes :

1. Arithmétique à deux facteurs

2. Conversion MØT

1.

Si $\left\{ \begin{array}{l} \langle \text{expqua } 1 \rangle \\ \langle \text{identificateur } 1 \rangle \\ \langle \text{constante } 1 \rangle \end{array} \right\}$ et $\left\{ \begin{array}{l} \langle \text{expqua } 2 \rangle \\ \langle \text{identificateur } 2 \rangle \\ \langle \text{constante } 2 \rangle \end{array} \right\}$ désignent

l'un et l'autre des mots; soit $\langle \text{terme } 1 \rangle$ et $\langle \text{terme } 2 \rangle$ les deux facteurs ci-dessus respectivement; alors :

$$\langle \text{terme } 1 \rangle \square \text{ PUIS } \square \langle \text{terme } 2 \rangle$$

est un terme qui désigne le mot formé du mot désigné par $\langle \text{terme } 1 \rangle$ suivi de celui désigné par $\langle \text{terme } 2 \rangle$, e.g.

$$\langle \text{terme } 1 \rangle \square \text{ PUIS } \square ' \square 120.85 ''$$

désigne le mot

$$\text{TOTAL} \square = \square 120.85$$

si $\langle \text{terme } 1 \rangle$ désigne le mot TOTAL $\square =$

$$G (G '' \square \text{ PUIS } \square ' \text{ABC} G) G ''$$

désigne le mot 'ABC'' (alors que 'ABC'' désigne le mot ABC).

C'est l'opération de concaténation

2. Conversion

Le format général est :

$$\text{MOT} \square \left\{ \begin{array}{l} \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\} \left[\square \left\{ \begin{array}{l} \text{INFER} \\ \text{PROCHE} \\ \text{SUPER} \end{array} \right\} \right] \\ \langle \text{constante} \rangle \end{array} \right\}$$

Soit $\langle \text{terme } 3 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

désignant un nombre naturel, un nombre entier, un nombre décimal, un nombre réel ou un nombre booléen, alors :

$$MOT \square \langle \text{terme}_3 \rangle$$

est un terme qui désigne un mot, lequel n'est autre que la constante servant de nom propre à l'objet désigné par $\langle \text{terme}_3 \rangle$ e.g.

$$MOT \square \langle \text{terme}_3 \rangle \text{ désigne le mot } +128.05 \text{ si } \langle \text{terme}_3 \rangle \text{ désigne le nombre décimal } +128.05$$

Par convention

$$MOT \square \langle \text{terme}_3 \rangle \text{ désigne le mot } 010 \text{ si } \langle \text{terme}_3 \rangle \text{ désigne le nombre booléen .}$$

FAUX VRAI FAUX

Soit $\langle \text{terme}_4 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

désignant un index, alors :

$$MOT \square \langle \text{terme}_4 \rangle \left[\begin{array}{l} \left\{ \begin{array}{l} \text{INFER} \\ \text{PRØCHE} \\ \text{SUPER} \end{array} \right\} \end{array} \right]$$

est un terme qui désigne le mot représentant la distance qui sépare la position à laquelle se trouve placé l'index désigné par $\langle \text{terme}_4 \rangle$ et respectivement, la borne :

PREMIER

la plus voisine de PREMIER ou de DERNIER

DERNIER

PRØCHE est adopté en l'absence d'indication de borne.

2.7.7 - Termes et pseudos-termes désignant les positions d'un index

Ils sont de l'une des formes :

11.1. Double relatif

11.2. Simple relatif

11.3. Simple

5. Conversion

1. Si $\langle \text{terme}_1 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

désigne un nombre naturel et si $\langle \text{terme}_2 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

désigne une position d'un index, alors :

$$\langle \text{terme}_1 \rangle \square \left\{ \begin{array}{l} \text{AVANT} \\ \text{APRES} \end{array} \right\} \square \langle \text{terme}_2 \rangle$$

est un terme qui désigne une position d'index pour la même entrée de la même file auxquelles est associé $\langle \text{terme}_2 \rangle$.

2. Avec $\langle \text{terme}_1 \rangle$ défini comme ci-dessus :

$$\langle \text{terme}_1 \rangle \square \left\{ \begin{array}{l} \text{APRES} \square \text{PREMIER} \\ \text{AVANT} \square \text{DERNIER} \end{array} \right\}$$

est un **pseudo-terme**. Si le contexte permet de déterminer la file et l'entrée de cette file auxquelles il fait allusion, alors il est considéré comme un terme, étant entendu que PREMIER et DERNIER désignent respectivement la première et la dernière des positions possibles pour un index attaché à cette entrée de la file en question

3.

PREMIER

et DERNIER

sont aussi des **pseudos-termes**
e.g.

$$1 \sqcap \text{AVANT} \sqcap \langle \text{terme}_2 \rangle, 1 \sqcap \text{APRES} \sqcap \langle \text{terme}_2 \rangle$$

désignent respectivement la position qui précède et celle qui suit celle désignée par $\langle \text{terme}_2 \rangle$.

$$0 \sqcap \text{AVANT} \sqcap \langle \text{terme}_2 \rangle, 0 \sqcap \text{APRES} \sqcap \langle \text{terme}_2 \rangle$$

désignent la même position que $\langle \text{terme}_2 \rangle$

4. Soit $\langle \text{terme}_3 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

désignant un mot susceptible d'être interprété comme un **pseudo-terme** d'index, de l'une ou l'autre des formes suivantes nécessairement :

$$\langle \text{nombre naturel} \rangle \sqcap \left\{ \begin{array}{l} \text{APRES} \sqcap \text{PREMIER} \\ \text{AVANT} \sqcap \text{DERNIER} \end{array} \right\}$$

$$\text{ou} \left\{ \begin{array}{l} \text{PREMIER} \\ \text{DERNIER} \end{array} \right\}$$

alors :

$$\text{INDEX} \sqcap \langle \text{terme}_3 \rangle$$

est un **pseudo-terme** d'index

2.7.8 - Termes désignant des nombres booléens

Ils sont de l'une ou l'autre des formes :

8.1. $N\emptyset N$

8.2. $\emptyset U$

8.3. ET

Les facteurs sont les suivants :

1.

$$\text{Si } \left\{ \begin{array}{l} \langle \text{expqua1} \rangle \\ \langle \text{identificateur1} \rangle \\ \langle \text{constante1} \rangle \end{array} \right\} \text{ et } \left\{ \begin{array}{l} \langle \text{expqua2} \rangle \\ \langle \text{identificateur2} \rangle \\ \langle \text{constante2} \rangle \end{array} \right\} \text{ désignent}$$

des nombres booléens, soit $\langle \text{terme1} \rangle$ et $\langle \text{terme2} \rangle$ les deux facteurs ci-dessus respectivement, alors :

les termes désignant des nombres booléens sont de l'une ou l'autre des formes suivantes :

$$N\emptyset N \square \langle \text{terme1} \rangle$$

$$\langle \text{terme1} \rangle \square \emptyset U \square \langle \text{terme2} \rangle, \quad \langle \text{terme1} \rangle \square \emptyset U \square N\emptyset N \square \langle \text{terme2} \rangle$$

$$N\emptyset N \square \langle \text{terme1} \rangle \square \emptyset U \square \langle \text{terme2} \rangle, \quad N\emptyset N \square \langle \text{terme2} \rangle \square \emptyset U \square N\emptyset N \square \langle \text{terme2} \rangle$$

$$\langle \text{terme1} \rangle \square ET \square \langle \text{terme2} \rangle, \quad \langle \text{terme1} \rangle \square ET \square N\emptyset N \square \langle \text{terme2} \rangle$$

$$N\emptyset N \square \langle \text{terme1} \rangle \square ET \square \langle \text{terme2} \rangle, \quad N\emptyset N \square \langle \text{terme1} \rangle \square ET \square N\emptyset N \square \langle \text{terme2} \rangle$$

Dans le cas où les nombres booléens désignés par $\langle \text{terme1} \rangle$ et $\langle \text{terme2} \rangle$ ne sont pas de même longueur, le plus court est supposé complété en tête par des éléments FAUX (0).

Les opérations sont effectuées sur les deux nombres booléens, élément par élément. Si A et B sont deux éléments correspondants, on applique les règles suivantes :

A	B	$\overline{A}$	$\overline{B}$	A $\cup$ B	A $\cap$ B	$\overline{A \cap B}$	$\overline{A \cup B}$	$\overline{A \cap B} \cup \overline{A \cup B}$
0	0	1	1	0		1	1	1
0	1	1	0	1		0	1	1
1	0	0	1	1		1	0	1
1	1	0	0	1		1	1	0

A	B	A $\cap$ B	A $\cup$ B	$\overline{A \cap B}$	$\overline{A \cup B}$	$\overline{A \cap B} \cup \overline{A \cup B}$
0	0	0		0		1
0	1	0		0		0
1	0	0		1		0
1	1	1		0		0

Soit $\langle \text{terme}_3 \rangle$ défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

désignant un nombre naturel ou un mot susceptible d'être interprété comme un nombre booléen ou un nombre naturel, alors :

$$\text{BOOLEEN} \sqsupset \langle \text{terme}_3 \rangle$$

est un terme désignant un nombre booléen.

Ainsi :

$$\text{BOOLEEN} \sqsupset 107$$

désigne le nombre booléen :

$$\text{VRAI} \sqsupset \text{VRAI} \sqsupset \text{FAUX} \sqsupset \text{VRAI} \sqsupset \text{FAUX} \sqsupset \text{VRAI} \sqsupset \text{VRAI}$$

car :

$$107 = (1 * 2^6) + (1 * 2^5) + (0 * 2^4) + (1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (1 * 2^0)$$

2.7.9 - Termes désignant des conditions

Ils sont de l'une ou l'autre des formes :

– ~~BOOLEEN~~ (voir 2.7.7)

9. Condition

Les termes précédents (2.7.7) désignant des nombres booléens peuvent aussi désigner des conditions: il en est de même d'une variable ou d'une constante qui désigne un nombre booléen

La condition est satisfaite si tous les éléments booléens sont vrais.

Dans le cas général, les facteurs sont les suivants :

$$s: \left\{ \begin{array}{l} \langle \text{expqua}_1 \rangle \\ \langle \text{identificateur}_1 \rangle \\ \langle \text{constance}_1 \rangle \end{array} \right\} \quad \text{et} \quad \left\{ \begin{array}{l} \langle \text{expqua}_2 \rangle \\ \langle \text{identificateur}_2 \rangle \\ \langle \text{constante}_2 \rangle \end{array} \right\} \quad \text{désignent}$$

tous les deux soit :

- des nombres naturels
- des nombres entiers
- des nombres décimaux
- des nombres réels
- des nombres booléens
- des mots
- des éléments de nature ADLIB
- des valeurs d'index attachées à la même entrée de la même file
- des parties d'articles ou de files placés en mémoire et composées exclusivement d'objets de l'une ou l'autre des natures ci-dessus.

Le format général devient :

$$\langle \text{terme}_1 \rangle \left\{ \begin{array}{c} - \\ < \\ < = \\ > \\ \backslash \\ \neq \end{array} \right\} \langle \text{terme}_2 \rangle$$

La forme $\langle \text{terme}_1 \rangle$ désignant un nombre booléen a été traitée ci-dessus.

Ce terme désigne une condition.

Lorsque $\langle \text{terme}_1 \rangle$ et $\langle \text{terme}_2 \rangle$ désignent des nombres non booléens, les remarques suivantes s'appliquent, elles sont d'ailleurs adoptées dans tous les termes explicités ci-dessus :

- Si $\langle \text{terme}_1 \rangle$ et $\langle \text{terme}_2 \rangle$ désignent des nombres de format différent, le plus court est amené au format du plus long de la façon suivante :

- pour les naturels avec des zéros à gauche
- pour les entiers avec des zéros à gauche (en fait propagation du signe)
- pour les décimaux avec les définitions suivantes :

considérons deux nombres décimaux de formats :

$$(p.q) \quad \text{et} \quad (r.s)$$

p, r nombres de chiffres des parties entières

q, s nombres de chiffres des parties décimales

Soit : $E = \max(q, r)$

la plus longue partie entière. Le nombre dont la partie entière est la plus petite est complété en tête par des zéros jusqu'à la longueur E .

Soit : $D = \max(q, s)$

la plus longue partie décimale, deux cas se présentent :

1. $E + D < 31$

Dans ce cas les parties décimales sont complétées à droite par des zéros éventuellement (le signe est conservé).

2. $E + D \geq 31$

La ou les parties décimales sont tronquées et, au besoin la plus courte est complétée par des zéros (le signe est conservé).

REMARQUE :

La comparaison de deux nombres décimaux de formats :

$$(31.0) \quad \text{et} \quad (0.31)$$

ne peut s'effectuer .

Dans le cas où l'un des deux facteurs, ou les deux, est une constante , le format standard adopté est le suivant :

$$(24.7)$$

- pour les nombres réels, le nombre en simple précision est complété par des zéros à droite.
- pour les nombres booléens, le plus court est complété par des éléments "FAUX" en tête.
- pour les mots, le plus court des deux est complété par des blancs () à droite.
- pour les éléments ADLIB, le plus court est complété à droite par des éléments nuls (en hexadécimal X'00').

Les comparaisons d'éléments de nature MØT sont faites après transcodage si un objet de nature ØRDRE est présent dans le programme. En effet, une comparaison d'objets de nature MØT est faite selon le critère de l'ordre lexicographique des caractères. Ainsi la comparaison :

< terme 1 > < terme 2 >

est satisfaite ou non selon que < terme 1 > précède ou non < terme 2 > dans l'ordre lexicographique standard (EBCDIC) ou dans celui spécifié par l'objet de nature ØRDRE.

Les comparaisons d'éléments de nature ADLIB sont faites suivant l'ordre lexicographique standard dans tous les cas.

Pour ce qui est des comparaisons de nombres booléens, tout se passe comme si elles concernaient les nombres naturels qui sont les images des nombres booléens. Ainsi la condition :

VRAI □ FAUX □ VRAI □ FAUX > FAUX □ VRAI □ VRAI □ VRAI

est satisfaite comme le serait celle portant sur les nombres naturels :

$(1 * 2^3) + (0 * 2^2) + (1 * 2^1) + 0 * 2^0 > (0 * 2^3) + (1 * 2^2) + (1 * 2^1) + (1 * 2^0)$

soit :

$10 > 7$

Les conditions portant sur les valeurs d'index attachés à la même entrées de la même file s'interprètent facilement en tenant compte de la relation.

PREMIER < DERNIER

Ainsi la condition :

10 □ AVANT □ DERNIER < 5 □ AVANT □ DERNIER

est satisfaite.

Lorsque les facteurs désignent des parties d'articles ou de file de même **structure séquentielle** et dont les éléments sont des objets simples: la condition est déterminée de proche en proche en considérant successivement, en fonction des règles énoncées jusqu'ici, les éléments du premier jusqu'au dernier.

2.7.10 - Termes de longueur

Ils sont de l'une des deux formes :

10.1. Arithmétique

10.2. File

Examinons successivement chacune d'elles.

2.7.10.1 - $\emptyset$ RD et PRE

Soit $\langle \text{terme}_1 \rangle$:

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

désignant un nombre naturel, un nombre entier, un nombre décimal ou un nombre réel.

– Dans le cas d'un nombre naturel ou d'un nombre entier α :

$$\emptyset\text{RD} \square \langle \text{terme}_1 \rangle$$

désigne le nombre (naturel) de chiffres décimaux nécessaires pour exprimer α ; et ce avec la définition suivante :

$$\emptyset\text{RD} \square \langle \text{terme}_1 \rangle = \emptyset\text{RD} (2^N \cdot 8 - 1) - 1$$

où N est le nombre de caractères (octets) utilisés pour la représentation du nombre α

La même convention vaut pour les descriptions (2.3).

– Dans le cas d'un nombre décimal α , $\text{ORD} \square \langle \text{terme}_1 \rangle$ désigne le nombre de chiffres décimaux de la partie entière de α

– Dans le cas d'un nombre réel α $\text{ORD} \square \langle \text{terme}_1 \rangle$ désigne la valeur maximale de l'exposant de α .

PRE ne s'applique qu'aux nombres décimaux et aux nombres réels :

$$\text{PRE} \square \langle \text{terme}_1 \rangle$$

désigne le nombre (naturel) maximal des chiffres décimaux nécessaires à, soit exprimer la valeur du nombre, soit exprimer la valeur de la mantisse.

Pour les nombres réels, selon le format (simple ou double précision), les valeurs 7 ou 17 sont seules possibles.

Pour les nombres décimaux, $\text{PRE} \square \langle \text{terme}_1 \rangle$ représente le nombre total de chiffres décimaux nécessaires à exprimer la valeur du nombre, partie décimale et partie entière.

2.7.10.2 - LØNG

Dans un terme $\text{LØNG} \square \lambda$, le facteur λ peut être :

1. une file,
2. un élément répété 2ème forme-
3. un élément doublement répété.

2.7.10.2.1 - FILES

Rappelons ce qu'on entend par entrée d'une file et longueur d'une entrée :

1. Soit la description de file :

AUXILIAIRE & FILE & ALPHA & 10, 20, 5 &

Les longueurs sont désignées par :

$$\text{LØNG} \square \langle \text{identificateur} \rangle$$

$\langle \text{identificateur} \rangle$ désignant une file. Ce terme désigne la longueur de la file, i.e. le nombre (naturel) d'éléments de l'ensemble des index de cette file. Ainsi, d'après la description ci-dessus :

$\text{LØNG} \square \text{ALPHA}$ désigne le nombre naturel $1000 = 10 * 20 * 5$.

2. Une autre forme possible est :

$$\text{LØNG} \square \langle \text{identificateur} \rangle (\langle \text{index long}_1 \rangle [\{ , \langle \text{index long}_2 \rangle \} \dots])$$

où $\langle \text{identificateur} \rangle$ est le nom d'une file, et $\langle \text{index long}_i \rangle$ est défini par :

$$\left(\begin{array}{c} < \text{expqua}_i > \\ < \text{identificateur}_i > \\ \\ \text{DE} \left\{ \begin{array}{c} < \text{expqua}_{i1} > \\ < \text{identificateur}_{i1} > \\ \text{PREMIER} \end{array} \right\} \square \text{JUSQUA} \square \left\{ \begin{array}{c} < \text{expqua}_{i2} > \\ < \text{identificateur}_{i2} > \\ \text{DERNIER} \end{array} \right\} \end{array} \right)$$

Expressions dans lesquelles $< \text{expqua} >$ ou $< \text{identificateur} >$ désigne une valeur d'index attaché à l'entrée considérée de la file.

Il est nécessaire, dans cette expression que l'un au moins des $< \text{index long} >$ ne soit pas de l'une ou l'autre des formes :

$$\left\{ \begin{array}{c} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\}$$

– Le caractère ou l'expression $\text{DE} \square \text{PREMIER} \square \text{JUSQUA} \square \text{DERNIER}$.

Ce terme désigne un nombre naturel représentant le nombre d'éléments de l'ensemble des index pour les entrées de la file dans lesquelles figure une expression qui n'est pas de la forme :

$$\left\{ \begin{array}{c} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\}$$

les valeurs prises sont :

– ou $\text{DE} \square \text{PREMIER} \square \text{JUSQUA} \square \text{DERNIER}$, la longueur de l'entrée correspondante.

– $\text{DE} \square \left\{ \begin{array}{c} < \text{expqua}_1 > \\ < \text{identificateur}_1 > \end{array} \right\} \square \text{JUSQUA} \square \left\{ \begin{array}{c} < \text{expqua}_2 > \\ \text{DERNIER} \end{array} \right\}$ le nombre

d'éléments existants entre les deux positions d'index désignés par les deux facteurs.

– $\text{DE} \square \text{PREMIER} \square \text{JUSQUA} \square \left\{ \begin{array}{c} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\}$ le nombre d'éléments existants entre

PREMIER et la position d'index désigné par le facteur.

Ainsi, considérons la description suivante :

AUXILIAIRE & FILE & ALPHA & 10, 10, 10 & PART & CATAL & 1 & I, J, K &&&&&&
 & 2 & L, M, N &&&&&&
 & 3 & Ø, P, Q & ;

(Les index sont décrits dans des phrases de type 2 avec un statut AUXILIAIRE et un support MEMOIRE)

Soient les valeur suivantes des index

I, L, Ø : 5 □ APRES □ PREMIER

J, M, P : 2 □ AVANT □ DERNIER

K, N, Q : 3 □ APRES □ PREMIER

Alors LØNG □ ALPHA désigne $1000 = 10 * 10 * 10$

et LØNG □ ALPHA (DE □ PREMIER □ JUSQUA □ I, . . , DE □
 Ø □ JUSQUA □ Q) désigne $180 = | 5-0+1 | * 10 * | 3-5+1 |$

2.7.10.2.2 - ELEMENT REPETE 2ème FORME

LØNG peut être utilisé, pour connaître la longueur d'une répétition de 2ème forme, le format du terme est alors :

$$LØNG \square \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\}$$

< expqua > ou < identificateur >, nomment tous les éléments répétés sans faire figurer le numéro d'un élément (voir 2.5.1.3).

Un tel terme désigne le nombre (naturel) d'éléments figurant avant l'occurrence du drapeau, ou la longueur maximale de la répétition si le drapeau ne figure pas

2.7.10.2.3 - ELEMENT DOUBLEMENT REPETE

Dans ce cas :

$$LØNG \square \left\{ \begin{array}{l} < \text{expqua} > \\ < \text{identificateur} > \end{array} \right\}$$

désigne le nombre (naturel) d'éléments simples qui figurent dans la répétition avec l'occurrence du drapeau, ou la longueur maximale si le drapeau ne figure pas

2.8 - INSTRUCTIONS

Une instruction n'a de signification que dans le contexte d'une procédure.

Lorsqu'on a énumérées les procédures auxquelles on fera appel, lorsqu'on a décrits, voire définis, les objets du traitement, il reste à compléter une procédure par la description d'un traitement, **suite d'instructions** qui doivent être exécutées dans un ordre déterminé.

Cet ordre est l'ordre même où les instructions figurent dans le traitement, sauf indication contraire. Une telle indication est donnée par une instruction (voir 2.8.2) qui désigne la prochaine instruction à exécuter.

Il faut donc pouvoir désigner des instructions - ou encore leur donner un nom -

Ces noms sont des identificateurs qui, du fait de leur association à des instructions, deviennent des étiquettes. Cette association a la forme :

< étiquette > : < instruction >

Il s'agit là d'une description et d'une définition et l'occurrence de l'étiquette y est **décrite** et **définie**. En toute autre occurrence dans la procédure la même étiquette est **employée**.

Une étiquette ne peut être le nom que d'une seule instruction, autrement dit, un même identificateur ne peut figurer qu'une fois en zone "étiquette" dans les phrases de type 6 d'un même programme.

De même les identificateurs placés en zone "valeur" d'une phrase de 5 dans la définition d'une variable de nature **ETIQUETTE** (et de statut **AUXILIAIRE**) doivent figurer une fois et une seule en tant qu'étiquettes dans les phrases de type 6 d'un programme.

Les étiquettes ne doivent figurer en aucun des endroits suivants :

- en zone "références" d'une phrase de type 1 ;
- en zone "nom" d'une phrase de type 2 *
- en zone "nom de partition" d'une phrase de type 3 ou 4
- en zone "nom" d'une phrase de type 3 ou 4
- en zone "nom" d'une phrase de type 5.

REMARQUE :

Dans le cas d'une description en phrase de type 2 :

AUXILIAIRE & ETIQUETTE & < identificateur > ;

< identificateur > doit avoir une occurrence, et une seule, en tant qu'étiquette dans la partie traitement. Ce type de déclaration a été introduit pour créer un type d'étiquette "obligatoire"

(Si < identificateur > n'est pas utilisé dans les phrases de type 6, en tant qu'étiquette évidemment, une erreur est détectée à la compilation).

La partie traitement d'une procédure est exprimée par des phrases de type 6, de format général :

$$\left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \left\{ \dots \right\} \right\} \right] \langle \text{instruction} \rangle ; \right]$$

en plaçant :

- dans la zone étiquette :
un identificateur qui devient ainsi le nom de l'instruction qui le suit immédiatement dans la zone instruction de la même phrase.

En plaçant devant une instruction plusieurs identificateurs différents, on peut donner plusieurs noms à une même instruction.

- dans la zone instruction :
les instructions, dans l'ordre où elles doivent être effectuées sauf indication contraire ; la dernière d'entre elles est l'instruction FIN dont l'exécution met fin au traitement (l'instruction FIN est obligatoire dans tout programme). Deux instructions consécutives sont séparées par un point virgule.

On classe les instructions, en six catégories :

- Instructions d'attribution,
- Instructions de saut,
- Instruction de conversion,
- Appels de procédures,
- Instructions de changement de support (Entrées Sorties),
- Macros-instructions du compilateur

On examine les quatre premières dans le paragraphe 2.8, les deux dernières dans les paragraphes 2.9 et 2.10.

2.8.1 - Instructions d'attribution dans une procédure

Les instructions d'attribution sont de la forme :

$$\langle \text{facteur 1} \rangle := \langle \text{facteur 2} \rangle$$

NOTE :

On peut mettre plusieurs instructions par carte et une instruction sur plusieurs cartes.

où

< facteur 1 > est un identificateur ou une expression qualifiée.

< facteur 2 > est une constante, un identificateur, une expression qualifiée ou un terme (pour les termes du type "conversion" voir 2.8.3).

Une telle instruction est abréviation de la phrase "Attribuer à l'objet désigné par < facteur 1 > la valeur de l'objet désigné par < facteur 2 > sans altérer celui-ci.

Les expressions < facteur 1 > et < facteur 2 > doivent satisfaire aux cinq conditions suivantes :

1. < facteur 1 > ne doit pas figurer dans la zone "nom" d'une phrase de type 5 : en effet, une telle phrase est une définition qui fixe de façon définitive la valeur d'un objet dans le contexte du programme.
2. < facteur 1 > ne peut être une donnée, ni < facteur 2 > un résultat :
En d'autres termes seuls sont autorisés les cas suivants :

	< facteur 1 >	< facteur 2 >
S		DONNEE
T		
A	MIXTE	MIXTE
T	RESULTAT	
U		
T	AUXILIAIRE	AUXILIAIRE

(Toute violation de ces conditions est dénoncée comme une erreur au moment de la compilation).

Comme on le verra plus loin (2.8.5), cette restriction jointe à quelques contraintes légères, supprime les inconvénients liés à l'appel par nom et à l'appel par valeur pour les arguments des procédures.

3. < facteur 1 > et < facteur 2 > doivent désigner des objets de même nature, sauf si une correspondance a préalablement été établie entre eux (voir 2.4.4).

Ces objets ne peuvent être que de l'une des natures suivantes :

BØØLEEN

DECIMAL

ENTIER

FILE

ARTICLE

INDEX

MØT

NATUREL

REEL

ADLIB

la valeur des objets de nature :

PRØCEDURE

PARTITIØN

ETIQUETTE

ne peut changer , étant donné qu'ils sont assimilés à des constantes

Les natures des objets désignés par < facteur 1 > et par < facteur 2 > peuvent être ARTICLE et FILE (réciproquement FILE et ARTICLE).

4. Le support des objets désignés par < facteur 1 > et < facteur 2 > ne peut être que MEMØIRE.

Dans les autres cas on doit utiliser les instructions de changement de support (2.9).

5. Les objets désignés par < facteur 1 > et < facteur 2 > doivent satisfaire certaines conditions concernant leur encombrement et éventuellement leur structure

On va les examiner, cas par cas, suivant la nature des objets :

1. Si $\langle \text{facteur}_1 \rangle$ et $\langle \text{facteur}_2 \rangle$ désignent tous les deux des mots, des nombres booléens, des nombres naturels ou des nombres entiers, l'instruction :

$$\langle \text{facteur}_1 \rangle := \langle \text{facteur}_2 \rangle$$

ne peut être exécutée que si la longueur de $\langle \text{facteur}_1 \rangle$ (i.e. nombre de caractères, nombre d'éléments booléens ou encore ordre de grandeur suivant le cas) est au moins égale à celle de $\langle \text{facteur}_2 \rangle$ (au moment de l'exécution).

Si elle lui est supérieure, l'exécution de l'instruction :

$$\langle \text{facteur}_1 \rangle := \langle \text{facteur}_2 \rangle$$

donne à $\langle \text{facteur}_1 \rangle$ une valeur qui est celle de $\langle \text{facteur}_2 \rangle$ ainsi modifiée :

- cadrée à gauche et complétée à droite par des blancs ou espaces ($\square$) si $\langle \text{facteur}_1 \rangle$ est de nature MØT.
- cadrée à droite et complétée à gauche par des éléments FAUX (0) si $\langle \text{facteur}_1 \rangle$ est de nature BØØLEEN
- cadrée à droite et complétée à gauche par des zéros si $\langle \text{facteur}_1 \rangle$ est de nature NATUREL ou ENTIER (le signe est conservé dans ce dernier cas).

2. Si $\langle \text{facteur}_1 \rangle$ et $\langle \text{facteur}_2 \rangle$ désignent tous les deux nombres décimaux, l'instruction :

$$\langle \text{facteur}_1 \rangle := \langle \text{facteur}_2 \rangle$$

ne peut être exécutée que si les parties entières et décimales des deux facteurs satisfont à des conditions précises. Pour les énoncer plus commodément, supposons que :

$\langle \text{facteur}_1 \rangle$ décrit un objet avec les longueurs (p.q)

$\langle \text{facteur}_2 \rangle$ décrit un objet avec les longueurs (r.s)

Les conditions sont alors les suivantes :

- de toutes façons $(p + q), r + s) < 31$

- Si $p < r$, l'instruction ne peut s'effectuer : cela équivaudrait à une troncation de la partie entière ; donc il faut $p \geq r$, la partie décimale de $\langle \text{facteur}_1 \rangle$ est celle de $\langle \text{facteur}_2 \rangle$ soit tronquée ($q < s$) conservée telle quelle ($q = s$) ou complétée à droite par des zéros ($q > s$) (le signe est conservé).

Si $p \geq r$, la partie entière de $\langle \text{facteur}_1 \rangle$ est celle de $\langle \text{facteur}_2 \rangle$ complétée à gauche par des zéros.

La situation est analogue à celle qu'on a rencontrée lors de la description des termes (2.7.4).

3. Si $\langle \text{facteur 1} \rangle$ et $\langle \text{facteur 2} \rangle$ désignent tous deux nombres réels, l'instruction :

$$\langle \text{facteur 1} \rangle := \langle \text{facteur 2} \rangle$$

s'effectue de la façon suivante :

- la caractéristique de $\langle \text{facteur 1} \rangle$ est celle de $\langle \text{facteur 2} \rangle$;
- la mantisse de $\langle \text{facteur 1} \rangle$ est celle de $\langle \text{facteur 2} \rangle$, modifiée de la façon suivante :
 - tronquée si $\langle \text{facteur 1} \rangle$ est du type simple précision et $\langle \text{facteur 2} \rangle$ du type double précision.
 - conservée telle quelle si les types de $\langle \text{facteur 1} \rangle$ et de $\langle \text{facteur 2} \rangle$ sont identiques.
 - complétée à droite par des zéros si $\langle \text{facteur 1} \rangle$ est du type double précision et $\langle \text{facteur 2} \rangle$ du type simple précision.

4. Si $\langle \text{facteur 1} \rangle$ et $\langle \text{facteur 2} \rangle$ désignent tous deux des index, ils doivent être rattachés à la même entrée de la même file. Toutefois, $\langle \text{facteur 2} \rangle$ peut être un pseudo-terme d'index (2.7.7), dans ce cas, la file et l'entrée correspondante sont celles de $\langle \text{facteur 1} \rangle$. Enfin l'instruction ne peut s'effectuer que si la valeur d'index désignée par $\langle \text{facteur 2} \rangle$ peut être représentée dans la zone prévue pour $\langle \text{facteur 1} \rangle$.
Ainsi :

Soit I un index attaché à la seconde entrée de la file F; l'instruction :

$$I := 2 \square \text{APRES} \square \text{PREMIER} ;$$

place l'index I sur la 3ème position de la seconde entrée de F.

Les instructions d'attribution :

$$I := N \square \text{APRES} \square I ;$$

$$\text{et } I := N \square \text{AVANT} \square I ;$$

peuvent s'exprimer sous la forme :

$$\text{AV} \square I \square \text{DE} \square N ;$$

$$\text{et } \text{AR} \square I \square \text{DE} \square N ;$$

respectivement ; et même si N désigne la constante de nombre naturel 1, sous la forme abrégée :

$$\text{AV} \square I ;$$

$$\text{AR} \square I ;$$

On remarquera que, dans de telles instructions, l est forcément de l'un des statuts :

MIXTE

ou AUXILIAIRE

5. Si $\langle \text{facteur } 1 \rangle$ et $\langle \text{facteur } 2 \rangle$ désignent des articles, ceux-ci doivent être structurés par la même partition, partition dont les éléments doivent être des objets simples fondamentaux :

~~BOOLEEN~~

NATUREL

ENTIER

MOT

DECIMAL

REEL

ou des INDEX

Chaque paire d'éléments mis ainsi en correspondance doit satisfaire aux conditions énoncées plus haut concernant les longueurs et aussi le rattachement aux files et entrées de file lorsqu'il s'agit d'index.

6. Si $\langle \text{facteur } 1 \rangle$ et $\langle \text{facteur } 2 \rangle$ désignent des files, celles-ci doivent satisfaire les conditions suivantes :
- elles doivent avoir un même nombres d'entrées ;
 - les longueurs des entrées correspondantes doivent être les mêmes ;
 - les articles de ces files doivent satisfaire les conditions énoncées ci-dessus.

Dans le cas où $\langle \text{facteur } 1 \rangle$ désigne un article et $\langle \text{facteur } 2 \rangle$ une partie de file, cette file ne doit contenir qu'un seul article, et les deux facteurs doivent satisfaire aux conditions ci-dessus.

7. Si $\langle \text{facteur } 1 \rangle$ désigne un nombre booléen et si $\langle \text{facteur } 2 \rangle$ est un terme désignant une condition, du type non booléen, $\langle \text{facteur } 1 \rangle$ reçoit l'une/ou l'autre des valeurs suivantes :

1. une suite d'éléments VRAI (1) sur toute la longueur de nombre booléen lorsque la condition est satisfaite.
2. une suite d'éléments FAUX (0) sur toute la longueur du nombre booléen dans le cas contraire

Cette convention diffère quelque peu de celle adoptée pour les nombres booléens, mais elle est utilisée pour transmettre facilement une condition dans un appel de procédure, où l'usage de termes est interdit

REMARQUE :

Cette remarque concerne le traitement des éléments répétés 2ème forme et des éléments doublement répétés. Dans le cas où un tel élément n'est pas de statut DØNNEE ou de statut AUXILIAI et ayant fait l'objet d'une définition ; l'utilisateur doit avoir la possibilité de déplacer le drapeau. Ceci est fait à l'aide des deux instructions introduites à propos du traitement des index :

$$\left\{ \begin{array}{l} \text{AV} \\ \text{AR} \end{array} \right\} \square \langle \text{expqua}_1 \rangle \left[\square \text{DE} \square \left\{ \begin{array}{l} \langle \text{expqua}_2 \rangle \\ \langle \text{identificateur}_2 \rangle \\ \langle \text{constante}_2 \rangle \end{array} \right\} \right]$$

où l'expression $\langle \text{expqua} \rangle$ qui désigne l'élément drapeau dans la répétition de 2ème forme ou dans la double répétition, est structurée de la façon suivante :

$$\langle \text{identificateur}_1 \rangle (\left\{ \begin{array}{l} \langle \text{expqua}_3 \rangle \\ \langle \text{identificateur}_3 \rangle \\ \langle \text{chaîne de caractères} \rangle \end{array} \right\}) \left[(\langle \text{condition} \rangle) \right] \square \text{DE} \square .$$

où

$\langle \text{identificateur}_1 \rangle$ est le nom de l'élément répété

$\langle \text{expqua}_3 \rangle$, $\langle \text{identificateur}_3 \rangle$ ou ' $\langle \text{chaîne de caractères} \rangle$ ' désigne le drapeau.

$\langle \text{expqua}_2 \rangle$, $\langle \text{identificateur}_2 \rangle$ ou $\langle \text{constante}_2 \rangle$ désigne un nombre naturel représentant le déplacement de l'indice de longueur de la répétition.

Ainsi, soit la partition PARTFLAG structurant l'article ARTFLAG

PARTFLAG & & & & MØT & 2 & FLAG &

& & FLAG, 20 & & MØT & 2 & SUITE &

& & 'AA', 10 & 4 & MØT & 2 & SEQ ;

FLAG $\square$ DE $\square$ ARTFLAG & 'BB' ;

alors les instructions :

AV □ SUITE (FLAG □ DE □ ARTFLAG) □ DE □ ARTFLAG □ DE □ 3 ;

AR □ SEQ (AA") □ DE □ ARTFLAG □ DE □ 2 ;

permettent de déplacer la position du drapeau de 3 éléments en avant (pour atteindre la longueur maximale) et de reculer (vers l'origine) la position du drapeau de 2 éléments. Les valeurs des nouvelles zones dans le cas AV sont remises à blanc (□), dans le cas AR la valeur des zones situées après le drapeau est laissée telle quelle, seule la position correspondant à l'ancien drapeau est remise à blanc (□). Tout ceci, dans la mesure où les bornes sont respectées. e.g.

Soit les valeurs :

SUITE (1) □ DE □ ARTFLAG : XX

SUITE (2) □ DE □ ARTFLAG : YY

SUITE (3) □ DE □ ARTFLAG : ZZ

SUITE (4) □ DE □ ARTFLAG : AA

drapeau : BB

et

SEQ (1) (1) □ DE □ ARTFLAG : 11

SEQ (2) (1) □ DE □ ARTFLAG : 22

SEQ (3) (1) □ DE □ ARTFLAG : 33

SEQ (4) (1) □ DE □ ARTFLAG : 44

SEQ (1) (2) □ DE □ ARTFLAG : 55

drapeau : AA

Après l'exécution des instructions :

AV □ SUITE (FLAG □ DE □ ARTFLAG) □ DE □ ARTFLAG □ DE □ 2 ;

et AR □ SEQ ('AA") □ DE □ ARTFLAG □ DE □ 3 ;

les valeurs deviennent :

SUITE (1) □ DE □ ARTGLAG : XX

SUITE (2) □ DE □ ARTFLAG : YY

SUITE (3) □ DE □ ARTFLAG : ZZ

SUITE (4) □ DE □ ARTFLAG : AA

SUITE (5) □ DE □ ARTFLAG : □□

SUITE (6) □ DE □ ARTFLAG : □□

drapeau : BB
la suite inchangée
et

SEQ (1) (1) □ DE □ ARTFLAG : 11

SEQ (2) (1) □ DE □ ARTFLAG : 22

drapeau : AA
la suite inchangée

2.8.1.1 - RECAPITULATION

Une instruction d'attribution qui ne satisfait pas l'une quelconque des précédentes conditions est dépourvue de signification: une erreur sera détectée à la compilation ou à l'exécution

En résumé les instructions d'attribution sont de l'une des formes suivantes :

$$\left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \right\} \dots \right] \right] \left\{ \begin{array}{l} \langle \text{identificateur} \rangle \\ \langle \text{expqua} \rangle \end{array} \right\} := \left\{ \begin{array}{l} \langle \text{identificateur} \rangle \\ \langle \text{expqua} \rangle \\ \langle \text{terme} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

$$\left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \right\} \dots \right] \right] \left\{ \begin{array}{l} \text{AV} \\ \text{AR} \end{array} \right\} \square \left\{ \begin{array}{l} \text{expqua} \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

$$\left[\text{DE} \square \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\} \right] ;$$

Comme dans la première version du langage, les instructions à plusieurs opérandes, e.g.

$$((A + B) * C + (- D)) * * 4$$

n'ont pas été introduites. Elles peuvent l'être sans difficulté d'ailleurs.

La définition de ces types d'expression demande une grande attention en ce qui concerne les règles arithmétiques adoptées pour l'évaluation des expressions ; en outre, avec la forme concise des instructions d'affectation du langage, le fossé n'est pas excessivement grand. Ces raisons ont limité le type des expressions à la forme simple, pour le stade actuel.

2.8.2 - Instructions de saut

Les instructions que contiennent les phrases de type 6 d'un programme sont exécutées séquentiellement, sauf indications contraires. Ces indications sont données par les **instructions de saut**. Elles sont de deux types :

1. Inconditionnel

2. Conditionnel

2.8.2.1 - INSTRUCTIONS DE SAUT INCONDITIONNEL

Elles sont de trois formes :

$$1. \left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \dots \right\} \right] \right] \text{VERS} \square \left\{ \begin{array}{l} \langle \text{étiquette} \rangle \\ \langle \text{expqua} \rangle \end{array} \right\} ;$$

où $\langle \text{expqua} \rangle$ désigne une étiquette.

Une telle instruction indique que la prochaine instruction à exécuter est celle dont le nom est $\langle \text{étiquette} \rangle$ ou qui est désignée par $\langle \text{expqua} \rangle$.
Ainsi :

AUXILIAIRE & ARTICLE & LABEL & ETIQ & ;

AUXILIAIRE & NATUREL & 1 & 2 & ;

ETIQ & & 10 & ETIQUETTE & 3 & LABEL ;

LABEL □ DE □ LABEL 1 & E1 , E2 , E3 , E4 , E5 , E6 , E7 , E8 , E9 , E10 ;

E10 : E2 : _____

E9 : _____

E8 : E3 : _____

E6 : E4 : PROC, (_____ ; _____ ; I) ;

VERS □ LABEL (I) □ DE □ LABEL1 ;

E5 : E7 : _____

E1 : _____

2. La seconde forme est :

VERS □ FIN ;

indique que l'exécution du programme est terminée (exécution de l'instruction FIN)

Dans l'exemple précédent, si on veut que les étiquettes E5 et E7, par exemple, correspondent à FIN, i.e. que VERS □ E2 ou VERS □ E6 □ correspondent à VERS □ FIN, il faut étiqueter l'instruction FIN du programme :

E6 : E2 : FIN ;

3. La troisième forme est :

ERREUR ;

indique que l'exécution du programme doit être stoppée (que la procédure ERREUR doit être exécutée

2.8.2.2 - INSTRUCTIONS DE SAUT CONDITIONNEL

Elles sont de deux formes :

$$1. \left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \right\} \dots \right] \right] \text{ SI } \square \langle \text{terme}_1 \rangle \square \text{ VERS } \square \left\{ \begin{array}{l} \langle \text{étiquette} \rangle \\ \langle \text{expqua} \rangle \end{array} \right\} ;$$

où $\langle \text{terme}_1 \rangle$ est un terme désignant une condition (2.7.9) et $\langle \text{expqua} \rangle$ désigne une étiquette du programme

Cette instruction indique que l'instruction :

$$\text{VERS } \square \left\{ \begin{array}{l} \langle \text{étiquette} \rangle \\ \langle \text{expqua} \rangle \end{array} \right\}$$

doit être exécutée si la condition désignée par $\langle \text{terme}_1 \rangle$ est satisfaite. Si elle ne l'est c'est l'instruction suivante qui est exécutée.

Une instruction de saut conditionnel telle que :

$$2. \left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \right\} \dots \right] \right] \text{ SI } \square \langle \text{terme}_1 \rangle \square \left\{ \begin{array}{l} \text{VERS } \square \text{ FIN} \\ \text{ERREUR} \end{array} \right\} ;$$

s'interprète de la même manière en tenant compte des indications du paragraphe 2.8.2.1.

Les instructions de saut n'ont pas évolué par rapport à la 1ère version du langage.

Les extensions peuvent porter surtout sur les instructions de saut conditionnel, e.g.

$$\left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \right\} \dots \right] \right] \text{ SI } \square \langle \text{terme}_1 \rangle \square \langle \text{instruction} \rangle ;$$

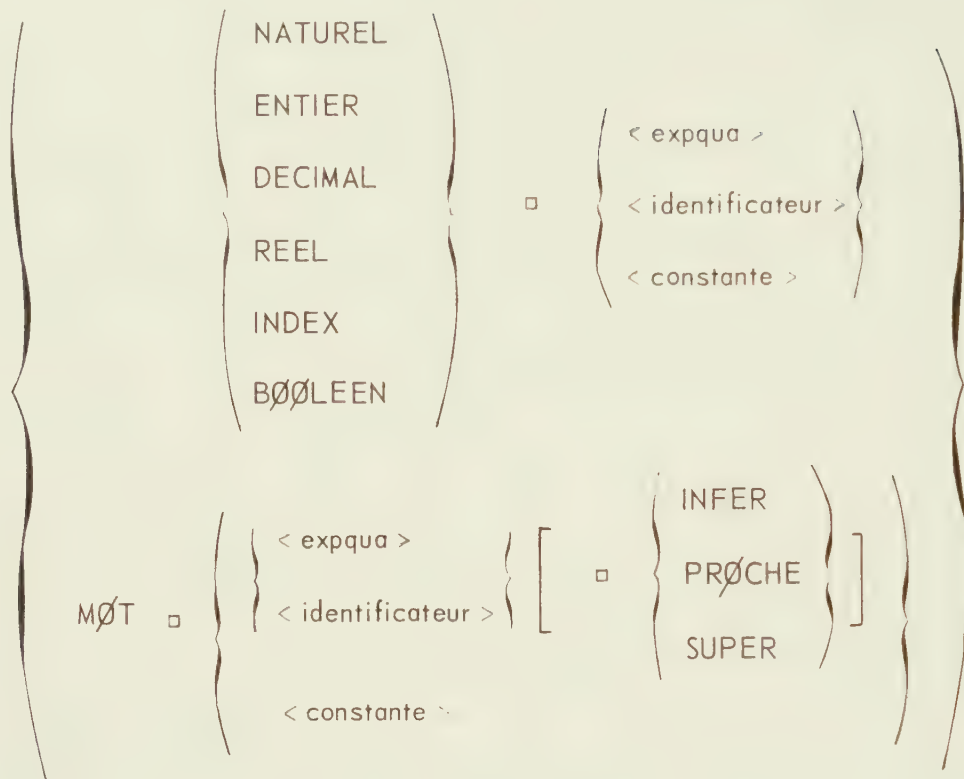
l'évolution sur la forme de $\langle \text{terme}_1 \rangle$ est liée à celles des expressions.

2.8.3 - Instructions de conversion

Les instructions de conversion, ou changement de nature, sont des instructions d'attribution dans lesquelles, le second facteur est un terme du type "conversion" (2.7.1). Le format général d'une telle instruction est :

$$\left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \right\} \dots \right] \right] \langle \text{facteur}_1 \rangle := \langle \text{facteur}_2 \rangle ;$$

où $\langle \text{facteur}_2 \rangle$ est défini par :



Le tableau suivant résume les possibilités pour les instructions de conversion.

Mot	Nature de l'objet figurant dans $\langle \text{facteur}_2 \rangle$						
Réservé	NATUREL	ENTIER	DECIMAL	REEL	BØØLEEN	INDEX	MØT
NATUREL	Ø	Ø	Ø	N	Ø	N	Ø
ENTIER	Ø	Ø	Ø	Ø	N	N	Ø
DECIMAL	Ø	Ø	Ø	Ø	N	N	Ø
REEL	Ø	Ø	Ø	Ø	N	N	Ø
BØØLEEN	Ø	N	N	N	Ø	N	Ø
INDEX	N	N	N	N	N	Ø	Ø
MØT	Ø	Ø	Ø	Ø	Ø	Ø	Ø

D'autres types de conversion sont possibles par l'intermédiaire des phrases de type 4

De plus, certaines conditions doivent être remplies :

1. de toute façon < facteur 1 > désigne un objet de même nature que le mot réservé figurant devant < facteur 2 >.
2. Comme pour les instructions d'attribution < facteur 1 > ne peut être une donnée, ni < facteur 2 > un résultat ; autrement dit, seuls sont autorisés les cas suivants :

	< facteur 1 >	< facteur 2 >
S T A T U T	MIXTE RESULTAT AUXILIAIRE	DONNEE MIXTE AUXILIAIRE

3. Les différents types de conditions sont pour chacune des conversions :

3.1 - NATUREL

La conversion NATUREL → NATUREL - conversion nulle - revient à une instruction d'affectation : les facteurs doivent simplement respecter les conditions énoncées plus haut.

La conversion ENTIER → NATUREL ne peut être effectuée que si l'objet désigné par < facteur 2 > est un nombre entier positif ou nul, et si il peut être représenté dans la zone affectée à < facteur 1 >.

La conversion DECIMAL → NATUREL ne peut être effectuée que si le nombre décimal désigné par < facteur 2 > est positif ou nul et si il peut être représenté dans la zone affectée à < facteur 1 >. Le nombre décimal est alors tronqué sur la position de la virgule

La conversion BOOLEAN → NATUREL ne peut être effectuée que si il est possible de représenter le profil binaire du nombre booléen dans la zone affectée à < facteur 1 >. Fonction de la longueur de cette zone, les valeurs maximales que peut prendre le nombre naturel désigné par < facteur 1 > sont :

1 octet	127	soit	0111 1111
2 octets	32 767	soit	0111 1111 1111 1111
4 octets	2 147 683 647	soit	0111 1111 1111 1111 1111 1111 1111 1111

La conversion $MOT \rightarrow NATUREL$ ne peut être effectuée que si l'objet désigné par $\langle \text{facteur } 2 \rangle$ est un mot qui peut être interprété comme une constante de nombre naturel et si la place réservée par $\langle \text{facteur } 1 \rangle$ est suffisante pour représenter ce nombre.

3.2 - ENTIER

La conversion $ENTIER \rightarrow ENTIER$ appelle les mêmes remarques que $NATUREL \rightarrow NATUREL$

La conversion $NATUREL \rightarrow ENTIER$ ne peut être effectuée que si il est possible de représenter l'objet désigné par $\langle \text{facteur } 2 \rangle$ dans la zone réservée à $\langle \text{facteur } 1 \rangle$. Avec les conventions de longueur un seul cas peut entraîner une difficulté, i.e. :

- $NATUREL$ représenté sur 4 octets et $ENTIER$ représenté sur 2 octets.

La conversion $DECIMAL \rightarrow ENTIER$ ne peut être effectuée que si il est possible de représenter la partie entière de l'objet désigné par $\langle \text{facteur } 2 \rangle$ dans la place réservée à $\langle \text{facteur } 1 \rangle$. La valeur de $\langle \text{facteur } 1 \rangle$ est celle de la partie entière de $\langle \text{facteur } 2 \rangle$ tronquée sur la position de la virgule. Le signe est conservé dans l'opération.

La conversion $REEL \rightarrow ENTIER$ ne peut être effectuée que si il est possible de représenter l'objet désigné par $\langle \text{facteur } 2 \rangle$ dans la place réservée à $\langle \text{facteur } 1 \rangle$. La valeur du nombre entier, résultat de la conversion, est la valeur de la partie entière tronquée à la position de la virgule du nombre réel.

La conversion $BOOLEEN \rightarrow ENTIER$ appelle les mêmes remarques que pour la conversion $BOOLEEN \rightarrow NATUREL$, sauf la limitation à 127 (il n'existe pas de nombres entiers représentés sur 1 octet).

La conversion $MOT \rightarrow ENTIER$ ne peut s'effectuer que si l'objet désigné par $\langle \text{facteur } 2 \rangle$ est un mot pouvant être interprété comme une constante de nombre entier et si la place réservée à $\langle \text{facteur } 1 \rangle$ est suffisante pour représenter ce nombre.

3.3 - DECIMAL

La conversion $NATUREL \rightarrow DECIMAL$ ne peut s'effectuer que si il est possible de représenter le nombre naturel désigné par $\langle \text{facteur } 2 \rangle$, dans la zone réservée à la partie entière du nombre décimal désigné par $\langle \text{facteur } 1 \rangle$. La partie décimale de ce nombre est remplie par des zéros.

La conversion $ENTIER \rightarrow DECIMAL$ appelle les mêmes remarques que ci dessus, en outre le signe est conservé dans l'opération

La conversion DECIMAL $\rightarrow$ DECIMAL ne peut s'effectuer que si il est possible de représenter la partie entière du nombre décimal désigné par < facteur 2 > dans la zone de la partie entière affectée à < facteur 1 >, cette partie étant complétée éventuellement par des zéros à gauche.

La partie décimale du nombre désigné par < facteur 2 > est rangée dans la zone affectée à la partie décimale de < facteur 1 >, cette partie est éventuellement tronquée, conservée telle quelle ou complétée à droite par des zéros. Le signe est conservé.

La conversion REEL $\rightarrow$ DECIMAL ne peut s'effectuer que si il est possible de représenter la partie entière du nombre réel désigné par < facteur 2 > dans la zone affectée à la partie entière du nombre décimal désigné par < facteur 1 >, cette partie est éventuellement complétée à gauche ou à droite par des zéros. La partie décimale du nombre décimal désigné par < facteur 1 > est remplie avec le reste de la mantisse (autrement dit : la partie décimale) du nombre réel tronquée, conservée telle quelle, ou complétée à droite par des zéros.

Le signe est conservé.

La conversion MØT $\rightarrow$ DECIMAL ne peut s'effectuer que si l'objet désigné par < facteur 2 > est un mot pouvant être interprété comme une constante de nombre décimal et si la place réservée à la partie entière du nombre par < facteur 1 > est suffisante pour recevoir ce nombre, complété éventuellement à gauche par des zéros, la partie décimale de la constante, si elle existe, est tronquée, conservée telle quelle ou complétée à droite par des zéros selon sa longueur vis à vis de celle de la partie décimale du nombre désigné par < facteur 1 >.

3.4 - REEL

La conversion NATUREL $\rightarrow$ REEL est toujours possible.

La conversion ENTIER $\rightarrow$ REEL est toujours possible.

La conversion DECIMAL $\rightarrow$ REEL est toujours possible.

La conversion REEL $\rightarrow$ REEL permet de passer du format simple précision au format double précision. Selon le cas la mantisse est tronquée à droite, conservée telle quelle ou complétée à droite par des zéros.

La conversion MØT $\rightarrow$ REEL ne peut s'effectuer que si l'objet désigné par < facteur 2 > est un mot pouvant être interprété comme une constante de nombre naturel. La mantisse du nombre est tronquée, conservée, telle quelle ou complétée à droite par des zéros selon le nombre de chiffres du mot et le format du nombre réel désigné par < facteur 1 >

3.5 - BØØLEEN

La conversion NATUREL $\rightarrow$ BØØLEEN ne peut s'effectuer que si le nombre naturel désigné par $\langle \text{facteur 2} \rangle$ n'est pas supérieur à la valeur maximale admissible pour un nombre booléen, compte tenu de sa longueur :

Longueur du booléen en octets	Valeur maximale du naturel
1	255
2	65 535
3	16 777 215
4	2 147 483 647

La conversion BØØLEEN $\rightarrow$ BØØLEEN ne peut s'effectuer que si la longueur du nombre booléen désigné par $\langle \text{facteur 2} \rangle$ est inférieure ou égale à celle du nombre booléen désigné par $\langle \text{facteur 1} \rangle$. Dans le cas où la longueur de $\langle \text{facteur 2} \rangle$ est inférieure, la valeur du nombre désigné par $\langle \text{facteur 1} \rangle$ est celle de celui désigné par $\langle \text{facteur 2} \rangle$ cadré à droite et complété à gauche par des éléments FAUX (0).

La conversion MOT $\rightarrow$ BØØLEEN ne peut s'effectuer que si l'objet désigné par $\langle \text{facteur 2} \rangle$ est un mot pouvant s'interpréter comme une constante de nombre booléen et que si la place réservée à $\langle \text{facteur 1} \rangle$ est suffisante pour recevoir ce nombre.

Un mot, pouvant s'interpréter comme une constante de nombre booléen, est l'une ou l'autre des formes :

$$\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} \left[\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} \dots \right]$$

$$\left\{ \begin{array}{c} \text{VRAI} \\ \text{FAUX} \end{array} \right\} \left[\left\{ \begin{array}{c} \text{VRAI} \\ \text{FAUX} \end{array} \right\} \dots \right]$$

– ou $\langle \text{constante de nombre naturel} \rangle$

3.6 - INDEX

La conversion INDEX $\rightarrow$ INDEX ne peut s'effectuer que si $\langle \text{facteur 1} \rangle$ et $\langle \text{facteur 2} \rangle$ désignent tous les deux des index attachés à la même entrée de la même file.

Il est aussi nécessaire que la zone réservée à l'objet désigné par $\langle \text{facteur 1} \rangle$ soit assez grande pour recevoir la valeur d'index désignée par $\langle \text{facteur 2} \rangle$.

La conversion MØT $\rightarrow$ INDEX ne peut s'effectuer que si $\langle \text{facteur 2} \rangle$ désigne un mot susceptible d'être interprété comme un pseudo-terme d'index (2.7.7), alors $\langle \text{facteur 1} \rangle$ désigne une valeur d'index, fixée par ce pseudo-terme d'index, attaché à une entrée d'une file.

3.7 - MØT

La conversion NATUREL $\rightarrow$ MØT ne peut s'effectuer que si $\langle \text{facteur 2} \rangle$ désigne un nombre naturel susceptible d'être représenté sous forme de caractères dans la zone affectée à l'objet désigné par $\langle \text{facteur 1} \rangle$. La longueur de cette zone doit être égale au moins à :

$$\emptyset RD \square \langle \text{facteur 2} \rangle + 1$$

Le mot ainsi formé est cadré à droite, et complété éventuellement par des blancs en tête, dans la zone définie depuis la gauche (de la zone affectée à l'objet désigné par $\langle \text{facteur 1} \rangle$) comme étant de longueur $\emptyset RD \square \langle \text{facteur 2} \rangle + 1$ complétée éventuellement à droite par des blancs.

La conversion ENTIER $\rightarrow$ MØT ne peut s'effectuer que si $\langle \text{facteur 2} \rangle$ désigne un nombre entier susceptible d'être représenté sous forme de caractères dans la zone affectée à $\langle \text{facteur 1} \rangle$. La longueur de cette zone doit être égale au moins à $\emptyset RD \square \langle \text{facteur 2} \rangle + 2$, une position étant conservée pour le signe du nombre.

Le mot ainsi formé est cadré à droite, dans la zone définie depuis la gauche (de la zone affectée à l'objet désigné par $\langle \text{facteur 1} \rangle$) comme étant de longueur $\emptyset RD \square \langle \text{facteur 2} \rangle + 2$ complétée éventuellement par des blancs à droite. Le signe du nombre (+ ou -) se trouve cadré à gauche dans $\langle \text{facteur 1} \rangle$, la zone éventuelle entre ce signe et le premier chiffre significatif étant remplie par des blancs.

REMARQUE :

Dans le cas où les valeurs des nombres naturels ou entiers sont nulles, seul un zéro subsiste en position des unités, le restant de la zone (à gauche) étant complété par des blancs (pour un nombre entier, le signe est omis).

La conversion DECIMAL $\rightarrow$ MØT ne peut s'effectuer que si $\langle \text{facteur 2} \rangle$ désigne un nombre décimal susceptible d'être représenté sous forme de caractères dans la zone affectée à $\langle \text{facteur 1} \rangle$. La longueur de cette zone doit être au moins égale à : $p + q + 2$ si $\langle \text{facteur 2} \rangle$ a fait l'objet d'une description (p.q).

Les deux positions supplémentaires sont attribuées pour le signe et le point décimal. Si la zone désignée par < facteur 1 > est plus longue, le mot obtenu est cadré à gauche (en longueur maximale) de façon inconditionnelle et complété à droite par des blancs. Le nombre décimal converti est placé dans la zone attribuée à < facteur 1 > cadré sur la position de la virgule définie par son format, le signe (+ ou -) est placé en tête et le point décimal (.) à la place de la virgule. Dans le cas où le nombre décimal est nul, il ne subsiste que les caractères '0.0' en tout et pour tout cadrés normalement.

Les parties entières et décimales étant cadrées sur la position de la virgule, la partie entière est complétée, à gauche, si nécessaire, par des blancs placés entre le signe et le premier chiffre significatif; la partie décimale conserve tous ses caractères, nuls ou non.

La conversion REEL $\rightarrow$ MØT ne peut s'effectuer que si < facteur 2 > désigne un nombre réel susceptible d'être représenté sous forme de caractères dans la zone affectée à l'objet désigné par < facteur 1 >. La longueur de cette zone doit être au moins égale à :

$$p + q + 5$$

si < facteur 2 > a fait l'objet d'une définition (p, q).

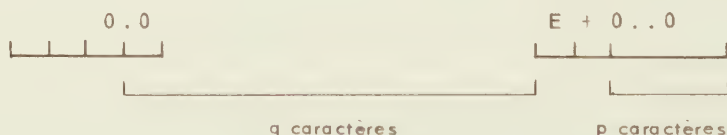
Le nombre réel converti, qui est de la forme :

$$\left\{ \begin{array}{c} + \\ - \end{array} \right\} 0 \quad q \text{ chiffres} > E \left\{ \begin{array}{c} + \\ - \end{array} \right\} < p \text{ chiffres} >$$

est placé avec ce format dans la zone désignée par < facteur 1 >; si cette dernière est plus longue, elle est complétée à droite par des blancs (□).

Dans le cas où le nombre de chiffres significatifs de la mantisse est inférieur à q, la mantisse est complétée à droite du dernier caractère nul par des blancs (□).

Un nombre réel nul s'exprime sous la forme



La conversion BOOLEEN $\rightarrow$ MØT ne peut s'effectuer que si < facteur 2 > désigne un nombre booléen susceptible d'être représenté sous forme de caractères dans la zone affectée à l'objet désigné par < facteur 1 >. La longueur de cette zone doit être au moins égale à la longueur du nombre booléen. En effet la forme résultant de ce type de conversion pour un nombre booléen, est :

$$\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} \left[\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} \dots \right]$$

Le nombre converti est placé dans la zone associée à $\langle \text{facteur 1} \rangle$ cadré à droite sur sa longueur maximale, cette dernière zone étant cadrée à gauche et complétée éventuellement à droite par des blancs ($\square$).

Aucun des caractères, quelle que soit sa valeur, n'est supprimé.

La conversion $\text{INDEX} \rightarrow \text{MOT}$ s'exprime par le format suivant :

$$\langle \text{facteur 1} \rangle := \text{MOT} \square \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\} \left[\square \left\{ \begin{array}{l} \text{INFER} \\ \text{PROCHE} \\ \text{SUPER} \end{array} \right\} \right]$$

Ce format avec les options INFER, PROCHE ou SUPER, est réservé au cas des conversions $\text{INDEX} \rightarrow \text{MOT}$.

En conséquence $\langle \text{expqua} \rangle$ ou $\langle \text{identificateur} \rangle$ désigne une position (ou valeur) d'un index attaché à une entrée d'une file.

Ce type de conversion permet d'exprimer la position d'un index.

INFER, PROCHE et SUPER indiquent respectivement quelle est la borne à choisir pour exprimer la position de l'index :

INFER : la position est exprimée par rapport à PREMIER

PROCHE : la position est exprimée par rapport à PREMIER ou à DERNIER suivant que l'index est plus proche du premier ou du dernier élément de l'entrée de la file. S'il est équidistant de ces deux éléments, la position est exprimée par rapport à PREMIER.

SUPER : la position est exprimée par rapport à DERNIER

PROCHE est adopté en l'absence d'indication de borne.

Le résultat de cette conversion est un mot de l'un des quatre formats suivants :

$\langle \text{nombre naturel 1} \rangle \square \text{ APRES } \square \text{ PREMIER}$

$\langle \text{nombre naturel 1} \rangle \square \text{ AVANT } \square \text{ DERNIER}$

PREMIER

DERNIER

$\langle \text{nombre naturel 1} \rangle$ étant un nombre naturel de 5 ou 10 chiffres selon la description qui a été faite de l'index dans les phrases de type 2, 3 ou 4 du programme. Il désigne la position de l'index par rapport à la borne exprimée ; sa représentation obéit aux mêmes règles que

Toutefois, lorsque le résultat de l'opération de conversion est

0 □ APRES □ PREMIER

ou 0 □ AVANT □ DERNIER

les expressions respectivement équivalentes :

PREMIER

DERNIER

soit substituées aux précédentes.

Les longueurs minimales acceptables pour < facteur 1 > sont de deux sortes :

1. Index du type 10 chiffres décimaux : 24 caractères

2. Index du type 5 chiffres décimaux : 19 caractères

En conséquence si la longueur de la zone associée à < facteur 1 > est inférieure, selon le cas, à 19 ou 24 caractères, le programme ne peut pas être exécuté.

< nombre naturel 1 > est cadré à gauche de la zone associée à < facteur 1 > et, dans celle-ci, à droite de la zone définie par la longueur affectée à l'index (5 ou 10 caractères) (les zones en tête de < nombre naturel 1 > sont remplacées par des blancs) ; viennent ensuite deux zones de longueur constante .

APRES □ PREMIER

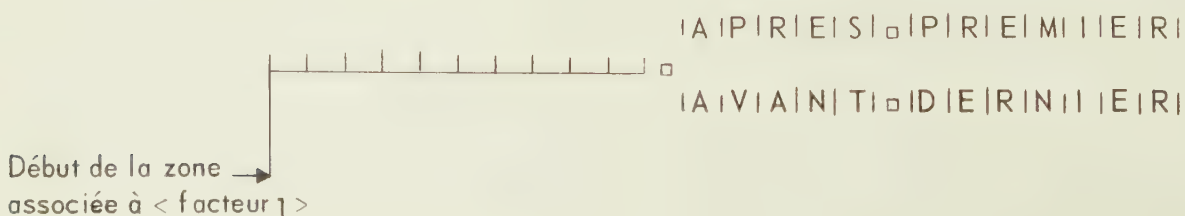
ou AVANT □ DERNIER

le restant de la zone, éventuel, étant rempli par des blancs.

e.g. Index à "5 caractères"



Index à "10 caractères"



REMARQUE :

Lorsque le résultat de la conversion est PREMIER ou DERNIER, ce mot est cadré à droite, et complété à gauche par des blancs, dans la zone qui serait normalement occupée par une expression complète i e



La conversion $M\emptyset T \rightarrow M\emptyset T$ ne peut s'effectuer que si $\langle \text{facteur 2} \rangle$ désigne un mot dont la longueur est inférieure ou égale à celle du mot désigné par $\langle \text{facteur 1} \rangle$.

La valeur de $\langle \text{facteur 1} \rangle$ est celle de $\langle \text{facteur 2} \rangle$ cadrée à gauche et complétée éventuellement à droite par des blancs ($\square$).

2.8.4 - Conversions implicites

Ce sont des instructions d'attribution entre objets, groupés ou non, placés en mémoire et appartenant à des articles ou à des files (ou étant des articles et des files) dont les partitions ont été décrites en parallèle.

Le format général est :

$$\left[\text{étiquette} > : \left[\left\{ \langle \text{étiquette} > : \right\} \dots \right] \right] \langle \text{facteur 1} \rangle := \langle \text{facteur 2} \rangle ;$$

ou $\langle \text{facteur 1} \rangle$ et $\langle \text{facteur 2} \rangle$, respectivement définis par :

$$\left\{ \begin{array}{l} \langle \text{expqua 1} \rangle \\ \langle \text{identificateur 1} \rangle \end{array} \right\} \text{ et } \left\{ \begin{array}{l} \langle \text{expqua 2} \rangle \\ \langle \text{identificateur 2} \rangle \end{array} \right\}$$

désignent tous les deux des éléments d'articles ou de files, ou encore des parties d'articles ou de files, ou encore des articles de support MEMOIRE, dont les partitions ont été décrites en parallèle (2.4.4).

La description des partitions en parallèle implique une suite de processus de conversion. Les conditions générales que doivent remplir les objets de ces conversions sont exposées au paragraphe précédent à propos des instructions de conversion (2.8.3).

2.8.5 - Appels de procédure

La forme générale d'un appel de procédure est :

$$\left[\begin{array}{l} \langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \right\} \dots \right] \end{array} \right] \left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\} , (\left[\langle \text{argd} \rangle \left[\left\{ \right\} , \langle \text{argd} \rangle \left\{ \right\} \dots \right] \right] ; \left[\langle \text{argm} \rangle \left[\left\{ \right\} , \langle \text{argm} \rangle \left\{ \right\} \dots \right] \right] ; \left[\langle \text{argr} \rangle \left[\left\{ \right\} , \langle \text{argr} \rangle \left\{ \right\} \dots \right] \right]) ;$$

où :

— $\langle \text{expqua} \rangle$ ou $\langle \text{identificateur} \rangle$ désigne - ou est - le nom de la procédure appelée.

Comme on l'a dit précédemment seuls les 8 premiers caractères du nom de la procédure sont pris en considération, le reste étant traité comme un commentaire. Ce nom de procédure ne peut commencer par l'un des deux mots réservés :

TRAITEME[NTIMMEDIAT]

ou MACRØ

Le nom de la procédure ainsi appelée doit être cité dans les phrases de type 1 du programme.

$\langle \text{argd} \rangle$ est un élément de la suite des arguments du type DØNNEE de l'appel,
 $\langle \text{argm} \rangle$ et $\langle \text{argr} \rangle$ sont des éléments des suites des arguments des types MIXTE et RESULTAT respectivement.

Soit en détaillant :

$\langle \text{argd} \rangle$ est défini par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \\ \langle \text{constante} \rangle \end{array} \right\}$$

$\langle \text{argm} \rangle$ et $\langle \text{argr} \rangle$ sont définis par :

$$\left\{ \begin{array}{l} \langle \text{expqua} \rangle \\ \langle \text{identificateur} \rangle \end{array} \right\}$$

En effet il n'est possible de faire figurer une constante que dans la suite des arguments ~~D~~ONNEE d'un appel de procédure ; il est donc impossible de modifier une constante par un appel de procédure puisqu'un argument ~~D~~ONNEE dans une procédure ne peut figurer :

- ni à gauche d'une instruction d'attribution, (ou comme facteur RESTE en division décimale),
 - ni parmi les arguments MIXTE et RESULTAT d'un appel de procédure.
- en d'autres termes, en aucun des endroits où sa valeur peut être modifiée.

Les conditions générales que doit remplir une instruction d'appel de procédure sont les suivantes :

1. Il doit y avoir une correspondance terme à terme entre les objets désignés par les éléments des suites ~~D~~ONNEE, MIXTE et RESULTAT de l'instruction d'appel et les objets de statut ~~D~~ONNEE, MIXTE et RESULTAT de la procédure appelée, l'ordre dans lequel ces objets sont énumérés dans les phrases de type 2 étant respecté.

En particulier, les objets groupés doivent être structurés par des partitions identiques, les longueurs - dans le cas où elles sont explicites et constantes - doivent être identiques. (Si elles sont omises, les longueurs des arguments sont utilisées pour compléter les informations fournies à la procédure appelée).

2. Une sous-partition ne peut figurer que dans les arguments ~~D~~ONNEE d'un appel de procédure.
3. Compte-tenu des conditions dans lesquels va se dérouler l'exécution de la procédure appelée et des effets qu'elle va produire :
 - les éléments de la suite des arguments "~~D~~ONNEE" ne peuvent pas désigner des objets de statut résultat mais seulement des objets de l'un des statuts :

~~D~~ONNEE

MIXTE

AUXILIAIRE

- les éléments de la suite des arguments MIXTE ne peuvent pas désigner des objets de statut ~~D~~ONNEE ou RESULTAT, mais seulement des objets de l'un des statuts :

MIXTE

AUXILIAIRE

- les éléments de la suite des arguments RESULTAT ne peuvent pas désigner des objets de statut ~~D~~ONNEE, mais seulement des objets de l'un des statuts :

MIXTE

RESULTAT

AUXILIAIRE

Ces dernières conditions sont résumées dans le tableau suivant :

	Type de la suite d'arguments		
	DØNNEE	MIXTE	RESULTAT
Statut de l'objet désigné par un élément de la suite	DØNNEE		
	MIXTE	MIXTE	MIXTE RESULTAT
	AUXILIAIRE	AUXILIAIRE	AUXILIAIRE

4. Un même argument (expression qualifiée ou identificateur) ne peut figurer à la fois dans deux des suites d'arguments DØNNEE, MIXTE et ou RESULTAT d'un appel de procédure. Une telle liberté risquerait en effet de faciliter de graves erreurs.

Cependant il est possible de faire figurer plusieurs fois un même argument dans une même suite DØNNEE, MIXTE ou RESULTAT

5. En aucun cas l'un des éléments désignés par les arguments de l'une des suites DØNNEE, MIXTE ou RESULTAT ne peut être de nature ETIQUETTE.

En effet, les étiquettes sont des objets internes à un programme et comme telles ne sont connues que du programme ; une procédure ne peut connaître les étiquettes du programme qui l'appelle et, réciproquement, ce programme ne peut pas connaître les étiquettes de la procédure qu'il appelle.

NOTE :

Une procédure peut s'appeler elle-même ; bien entendu, l'appel de cette procédure dans elle-même doit évidemment respecter les règles ci-dessus.

Cette liberté ne soulève aucune difficulté, les arguments (leurs adresses) et les auxiliaires étant construits dynamiquement à chaque appel de procédure. La zone allouée aux auxiliaires d'une procédure est générée à chaque appel, la place est restituée à la rencontre de l'instruction FIN.

Une fois exécutée l'instruction FIN dans la procédure appelée, l'instruction qui suit l'appel de procédure dans la procédure appelante est exécutée.

Du point de vue du traitement, une procédure se présente donc comme une instruction ; l'utilisateur a la possibilité de demander au compilateur d'insérer la procédure (voir 2.10.1.1).

Dans les appels de procédure, la transmission des paramètres est faite uniquement par le tranchement de leurs adresses, cela ne risque pas d'introduire d'erreurs du fait de la division des arguments en trois classes : DØNNEE, MIXTE et RESULTAT.

REMARQUE :

Les objets de statut AUXILIAIRE qui ont fait l'objet d'une définition dans un programme sont considérés, à l'égard du traitement, comme des objets de statut DØNNEE.

2.9 - INSTRUCTIONS DE CHANGEMENT DE SUPPORT

2.9.1 - Introduction - Le concept de macro-instruction

Dans la première version du langage, les opérations d'entrée-sortie étaient exprimées sous forme d'une instruction d'affectation.

C'est là une solution idéale, mais qui suppose un système d'exploitation spécialement adapté à ces opérations d'échange.

Nous avons le choix entre deux solutions :

- ou bien utiliser la forme simple, décrite dans la première version du langage, des opérations d'échange et réaliser un sous-système d'exploitation (fonctionnant sous le contrôle du système d'exploitation de l'ordinateur) pour l'interprétation de ces ordres,
- ou bien définir de nouveaux ordres mieux adaptés aux conditions réelles.

Nous avons retenu la seconde solution ; mais avant de commencer toute description, il nous a semblé utile de dégager quelques notions générales concernant le traitement des files placées sur des supports autres que MEMØIRE.

Le rapprochement des conclusions de cette analyse et des possibilités de l'Øperating System IBM 360 a permis de définir l'introduction des nouveaux ordres d'entrée-sortie dans le langage.

Nous débuterons cet exposé par des notions d'organisation des files sur les supports autres que MEMØIRE.

Les différents supports envisagés dans l'application présente peuvent se répartir en deux classes suivant qu'un article d'une file placée sur l'un de ces supports peut-être adressé ou non.

- Support adressable :

STØCK	c'est-à-dire	DISQUE
		TAMBØUR
		etc...

- Supports non adressables :

- BANDE
- Tous les supports externes

Sur les supports non adressables les articles des files y sont rangés d'une façon physique plutôt que logique, les uns à la suite des autres.

La position de l'enregistrement qui suit un article donné est l'emplacement physique suivant.

Cette organisation, dite séquentielle, est la seule pratiquement réalisable sur de tels supports.

Sur les supports adressables, par contre, la faculté de pouvoir désigner l'adresse de l'enregistrement auquel on veut accéder permet d'envisager d'autres organisations que l'organisation séquentielle.

On peut imaginer d'identifier un article d'une file (à une entrée évidemment) placée sur tel support en fournissant uniquement la valeur d'un index associé à cette file ou une valeur particulière associée à une variable contenue dans l'article.

Si cette opération est la seule possible sur une file, ceci oblige à une grande complication de la gestion d'accès aux articles ; en effet il faut, à ce moment, posséder une table contenant toutes les adresses des articles en fonction des différentes valeurs de l'index ou de la variable servant à identifier un article particulier. Cette table est un répertoire d'adresses (c'est-à-dire qu'elle permet de réaliser un adressage indirect).

Avec une telle organisation, les articles peuvent être placés dans un ordre aléatoire sur le support et une consultation purement séquentielle de cette file sera fort longue.

Aussi peut-on dériver de cette organisation, connue sous le nom d'organisation directe, plusieurs organisations permettant un emploi plus commode et plus efficace des possibilités d'adressage du support.

Une file peut être consultée tantôt séquentiellement ou tantôt dans un ordre aléatoire. Il faut, dans ce cas, éviter de pénaliser, par un temps de traitement trop long, l'utilisateur désirant consulter séquentiellement cette file.

Il est nécessaire que les articles composant la file soient placés séquentiellement sur le support, soit d'une façon physique ou d'une façon logique à l'aide de chainage entre les articles. Dans ce cas, il est possible de constituer un répertoire simplifié des adresses des articles en regroupant les valeurs des index ou des variables servant à identifier des articles particuliers en un répertoire simplifié.

On arrive ainsi à la notion connue sous le nom d'organisation séquentielle indexée.

L'utilisateur peut désirer traiter les articles d'une file tantôt séquentiellement ou tantôt séquentiellement à l'intérieur d'un ensemble identifié par une valeur (d'index ou de variable par exemple) particulière.

Par exemple, traitement séquentiel des articles d'une file compris entre les valeurs d'index :

30 □ APRES □ PREMIER
et 62 □ APRES □ PREMIER

Il est alors possible de considérer de tels groupements comme une partition de l'ensemble des éléments de la file.

On arrive ainsi à la notion connue sous le nom d'organisation partitionnée.

En résumé, nous avons vu trois façons d'organiser une file à une entrée placée sur un support du type **STØCK** :

- DIRECTE
- SEQUENTIELLE INDEXEE
- PARTITIONNEE

L'organisation séquentielle est évidemment réalisable sur les supports de type **STØCK**

Après avoir vu différentes organisations possibles pour des files placées sur des supports autre que **MEMØIRE**, nous devons examiner les ordres nécessaires au traitement de ces files.

Il importe, pour avoir un traitement commode des files, que les informations concernant leur emplacement, leur organisation, leurs caractéristiques particulières soient regroupées en une zone particulière destinée à réaliser l'interface entre l'utilisateur et le superviseur.

A cet effet on dispose d'un objet de nature particulière : la variable de contrôle des files.

L'utilisateur doit pouvoir rédiger un programme sans avoir à décrire avec précision le type et l'adresse du périphérique destinés à recevoir la file ; dans ce but la variable de contrôle des files contient une référence à un ordre du langage de commande du superviseur, cet ordre décrivant avec précision les paramètres obsents.

Cet ordre du langage de commande du superviseur est, dans le cas d'un système parfaitement adapté, une instruction particulière du programme de type **TRAITEMENT IMMEDIAT**.

L'un des points à examiner est le statut de la file qui est l'objet du traitement :

- DØNNEE** - la file ne peut qu'être lue,
- RESULTAT** - la file est à créer,
- MIXTE** - dans ce cas, on peut distinguer deux types de traitement :

- lecture puis écriture éventuelle d'un article - c'est le mode mise à jour - ,
- écriture puis lecture éventuelle d'un article - c'est le mode vidage temporaire -

Les organes périphériques travaillent à une cadence nettement plus faible que la mémoire centrale de l'ordinateur ; bien souvent le rapport entre le temps d'accès à une information placée en mémoire et celui relatif à une information placée sur un disque magnétique est de l'ordre de celui existant entre une seconde et plusieurs jours.

En conséquence, durant l'accès à un fichier, l'unité centrale de l'ordinateur peut ne rien avoir à faire d'autre que d'attendre les informations en provenance du périphérique concerné par l'opération. Deux solutions sont possibles :

- dans un contexte de multiprogrammation, le système d'exploitation utilisera ces temps d'attente pour faire exécuter, par l'unité centrale, une autre tâche et il rendra le contrôle au premier programme une fois que l'opération d'entrée-sortie sera terminée,
- même dans un contexte de multiprogrammation, l'utilisateur peut demander au système d'initialiser une opération d'échange, et, une fois cette opération "lancée", continuer l'exécution de son programme. L'utilisateur doit alors assurer lui-même la synchronisation de son traitement avec les opérations d'échange ; c'est-à-dire tester lui-même la fin des opérations d'échange au cours de son traitement.

Dans le premier cas, nous avons à faire à des méthodes d'accès aux files dites assistées (assistées par le système) et dans l'autre cas, à des méthodes d'accès non assistées.

Dans l'utilisation des méthodes d'accès assistées, le système d'exploitation délivre à l'utilisateur les articles qui constituent les files, puisqu'il lui est possible d'assurer certains traitements ; dans les méthodes d'accès non assistées, l'utilisateur reçoit l'image en mémoire de la représentation physique des articles sur le support considéré.

Ces deux types de traitement sont connus, plus généralement, sous le nom de travail en mode logique ou de travail en mode physique (ou mode "device").

Dans les méthodes d'accès assistées, un point d'importance est le traitement des articles en mémoire, ou plus exactement le mécanisme d'adressage grâce auquel le système d'exploitation délivre les articles de la file à l'utilisateur. Le système introduit plusieurs articles en mémoire à la fois et il réalise un traitement sur ces informations pour délivrer les articles à l'utilisateur. En d'autres termes, le système réalise la transformation entre le mode physique (informations) et le mode logique.

Cette transformation ne peut s'opérer que dans une zone déterminée gérée par le système (zone tampon ou buffer), en conséquence si l'utilisateur désire obtenir l'article dans une zone appartenant à son programme, l'article doit être déplacé par le système ; une autre technique consiste à demander au système de fournir, non par un article dans une zone particulière, mais plutôt

l'adresse de l'article dans la zone tampon gérée par le système ; en d'autres termes le système retourne à l'utilisateur la valeur d'un index de la file constituée par les zones tampon.

C'est la technique de base ; à partir de cela on peut imaginer des jeux plus subtils pour des opérations particulières sur des files traitées dans les modes désignés, mais le traitement des zones tampons se ramène au cas précédents et n'est donc qu'une extension.

Après avoir introduit, de façon dialectique certes, les différentes organisations et les différents modes de traitement des files ; il nous faut à présent décrire les ordres nécessaires à ce traitement.

1 - Files d'organisation séquentielle

1.1 - Méthode assistée

Lecture d'un article

Ecriture d'un article

Réécriture d'un article mis à jour

1.2 - Méthode non-assistée

Lecture d'un bloc (article physique)

Ecriture d'un bloc

Attente de la fin d'une opération

2 - Files d'organisation séquentielle indexée

2.1 - Méthode assistée

Constitution d'une telle file, écriture

Lecture d'un article

Réécriture d'un article mis à jour

Début de la recherche séquentielle des articles

Fin de la recherche séquentielle des articles

2.2 - Méthode non-assistée

Lecture d'un bloc

Ecriture d'un bloc

Ecriture d'un bloc

Attente de la fin d'une opération

3 - Files d'organisation partitionnée

3.1 - Méthode assistée

Constitution d'un nouveau nom de groupe d'articles

Recherche d'un groupe d'articles de nom donné

Lecture d'un article
 Ecriture d'un article
 Réécriture d'un article mis à jour

3.2 - Méthode non-assistée

Constitution d'un nouveau nom de groupe d'articles
 Recherche d'un groupe d'articles de nom donné
 Lecture d'un bloc
 Ecriture d'un bloc
 Attente de la fin d'une opération

4 - Files d'organisation directe

4.1 - Méthode assistée

Ecriture d'un article
 Lecture d'un article
 Ecriture d'un article mis à jour

4.2 - Méthode non-assistée

Lecture d'un bloc
 Ecriture d'un bloc
 Attente de la fin d'une opération

Nous n'avons pas parlé des opérations communes aux différents traitements envisagés :

- ouverture d'une file
- fermeture d'une file
- positionnement
- contrôle des périphériques,
 etc...

Ces opérations sont toujours nécessaires, bien qu'elles soient souvent considérées comme implicites.

Le système de gestion de fichiers de l'Øperating System IBM 360 présente une grande quantité de points communs avec les notions que nous venons d'exposer, aussi avons nous choisi d'introduire les facilités offertes, par l'ØS 360, dans le langage. Cette introduction ne pose guère de problèmes, puisque les types de traitement des files dans le langage et dans l'ØS 360 présentent de grandes ressemblances, le formalisme étant toutefois différent

Cette introduction décidée, restait à définir l'expression de ces opérations de changement de support dans le langage.

Il nous a semblé, en premier lieu, qu'une telle opération pouvait fort bien s'apparenter à un appel de procédure, e.g. :

LIRE, (< nom de file > ; < index > ; < article lu >) ;

mais certaines opérations mettent en jeu des caractéristiques locales du programme - par exemple des étiquettes correspondant à des séquences d'instructions réalisant un traitement particulier -, ces caractéristiques ne peuvent pas figurer dans un appel de procédure tel qu'il est défini dans le langage.

Une solution consistait alors à exprimer cette opération en plusieurs instructions, mais elle présentait l'inconvénient d'exprimer difficilement une opération pourtant fort naturelle.

Comme ces opérations ne peuvent s'exprimer d'une façon générale, à l'aide d'un appel de procédure, on a été conduit à introduire la notion de macro-instruction dans le langage.

La forme de ces macro-instructions s'inspire de celle des appels de procédure.

Le format général est le suivant :

$$\left[\langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \{ \dots \} \right\} \right] \text{MACRØ} \langle \text{type macro} \rangle, \langle \text{partie procédure} \rangle \right. \\ \left. \left[, \langle \text{partie macro} \rangle \right] ; \right]$$

où

- < type macro > est un identificateur, appartenant à une liste préétablie, qui désigne le type de la macro-instruction ;
- < partie procédure > est définie par :

$$\left(\left[\langle \text{argd} \rangle \left[\left\{ , \langle \text{argd} \rangle \{ \dots \} \right\} \right] ; \left[\langle \text{argm} \rangle \left[, \langle \text{argm} \rangle \{ \dots \} \right] ; \left[\langle \text{argr} \rangle \left[\left\{ , \langle \text{argr} \rangle \{ \dots \} \right\} \right] \right] \right) \right]$$

elle est identique à la séquence d'arguments d'un appel de procédure et suit les mêmes règles en ce qui concerne les types et les statuts des arguments.

Cette séquence d'arguments désigne les éléments employés dans l'opération exprimée par la macro-instruction < type macro >.

- < partie macro > est définie par :

$$\left\{ , \left(\langle \text{caractéristique} \rangle , \langle \text{spécification} \rangle \right) \right\} \dots$$

où **< caractéristique >** est un identificateur désignant une fonction particulière réalisée par la macro-instruction, et **< spécification >** est une suite d'éléments ayant trait à la fonction référencée par **< caractéristique >**.

Une telle suite regroupe :

- les objets qu'il est impossible de faire figurer dans un appel de procédure, donc dans **< partie procédure >** ;
- des éléments propres à la fonction désignée (ou spécifiée) par **< caractéristique >**.

Pour chaque type de macro-instruction, une forme particulière de **< partie macro >** est définie

Les macro-instructions introduites dans le langage se répartissent en trois classes :

- les macro-instructions relatives aux déclarations ou plus simplement macro-déclarations.

Ces macro-déclarations évitent de déclarer certains éléments sous la forme standard du langage. Il s'agit, par exemple, de la macro-instruction **PARAL** servant à mettre en parallèle deux descriptions de partition ou encore de la macro-instruction **DCB** servant à générer la table d'interface entre le système et l'utilisateur pour le traitement des files.

- les macro-instructions relatives à des traitements que doit exécuter le compilateur ou plus simplement macro-directives.

Il s'agit, par exemple, de la macro-instruction **INSERT** indiquant au compilateur qu'il doit substituer l'expression d'une procédure en lieu et place de son appel, ou encore des macro-instructions **CATAL** et **DELCATAL** qui permettent de gérer le catalogue des partitions.

- les macro-instructions relatives au traitement proprement dit, appelées macro-instructions en l'absence d'une terminologie plus appropriée.

Dans les paragraphes suivants, nous allons d'abord présenter sommairement les possibilités qu'offre l'Operating System IBM 360 dans ce domaine ; nous présenterons ensuite l'introduction de ces facilités dans le langage.

Le paragraphe 2.10 contient la description des macro-instructions du compilateur.

Avant d'aborder l'une et l'autre, rappelons brièvement les raisons qui ont guidé notre choix.

- les macro-instructions d'entrée-sortie de l'Øperating System IBM 360 sont relativement complexes, mais elles représentent, en fait quasiment toutes les instructions de changement de support qui nous sont nécessaires. Les introduire dans le langage permet de définir, par la suite, toute nouvelle forme pour les instructions de changement de support ; ces instructions se ramenant finalement aux macro-instructions de l'Øperating System.
- dans un centre de calcul, celui qui s'occupe du système d'exploitation - l'ingénieur "système" en général - peut, même sans une connaissance approfondie du langage, aider les utilisateurs en ce qui concerne cette partie entrée-sortie.
- dans le traitement des problèmes de gestion, la partie entrée-sortie d'un programme est rarement sujette à des modifications ; l'économie de temps de traitement réalisée par l'utilisation des ordres actuels compense largement le fait que l'écriture de cette partie a coûté un peu plus de temps.

2.9.2 - Organisation des fichiers

Les ensembles de données, ou fichiers, i.e. les files à une entrée (ou 2 pour certains supports), sont organisés sur les supports internes autres que MEMØIRE et sur les supports externes de quatre façons différentes:

2.9.2.1 - ORGANISATION SEQUENTIELLE

C'est la structure habituelle des files de support BANDE dans laquelle les articles sont placés d'une façon physique plus que logique : les uns à la suite des autres. La position de l'enregistrement suivant un article donné est l'emplacement physique suivant.

L'organisation séquentielle est utilisée pour tous les supports BANDE, elle peut être aussi spécifiée pour les périphériques à accès direct STØCK. Dans ce cas le mot désignant l'emplacement d'un fichier sur un support de la classe STØCK organisé séquentiellement devient :

STØCKSEQ

les supports externes :

ECARTE

ERUBAN

CLAVIER

SCARTE

SRUBAN

PAPIER

IMPRIME

ne peuvent recevoir que des fichiers organisés séquentiellement.

2.9.2.2 - ORGANISATION SEQUENTIELLE INDEXEE

Cette organisation est exclusivement réservée aux fichiers représentés sur des périphériques appartenant à la classe **STØCK**; le mot qui indique la place d'un fichier organisé en séquentiel indexé est :

STØCKIND

Il s'agit, dans la majorité des cas, de disques magnétiques. L'organisation séquentielle ou l'organisation partitionnée (9.2.4) ne permettent d'accéder facilement à un article particulier qu'après une lecture préalable des enregistrements précédents ou grâce à un enregistrement spécial et séparé, qui donne la position des articles et qu'on est obligé de conserver.

Au contraire, l'organisation séquentielle indexée permet d'éviter une telle pratique. En effet, dans ce cas, l'emplacement d'un article sur le support dépend du contenu d'une partie de l'article, la clé, et lorsque l'utilisateur donne la valeur de cette clé, le système peut trouver directement l'article.

Pour créer un tel fichier, il faut trier les articles sur la valeur de la clé. Cette suite ordonnée d'articles est alors écrite sur le support **STØCKIND**. Chaque fois qu'une piste est complète, la plus grande valeur de la clé sur cette piste est conservée sur une piste appelée Piste-Index (Track-Index). Lorsque toutes les pistes d'un cylindre sont complètes, la plus grande valeur de la clé sur le cylindre est conservée sur un cylindre appelé Cylindre-Index.

Pour lire un article, il suffit de fournir sa clé. Le système recherche dans le cylindre-Index le numéro du cylindre contenant la clé, une fois le cylindre atteint, la piste-index est explorée pour trouver le numéro de la piste qui contient l'article. Enfin, une recherche sur cette piste permet de trouver l'article désiré.

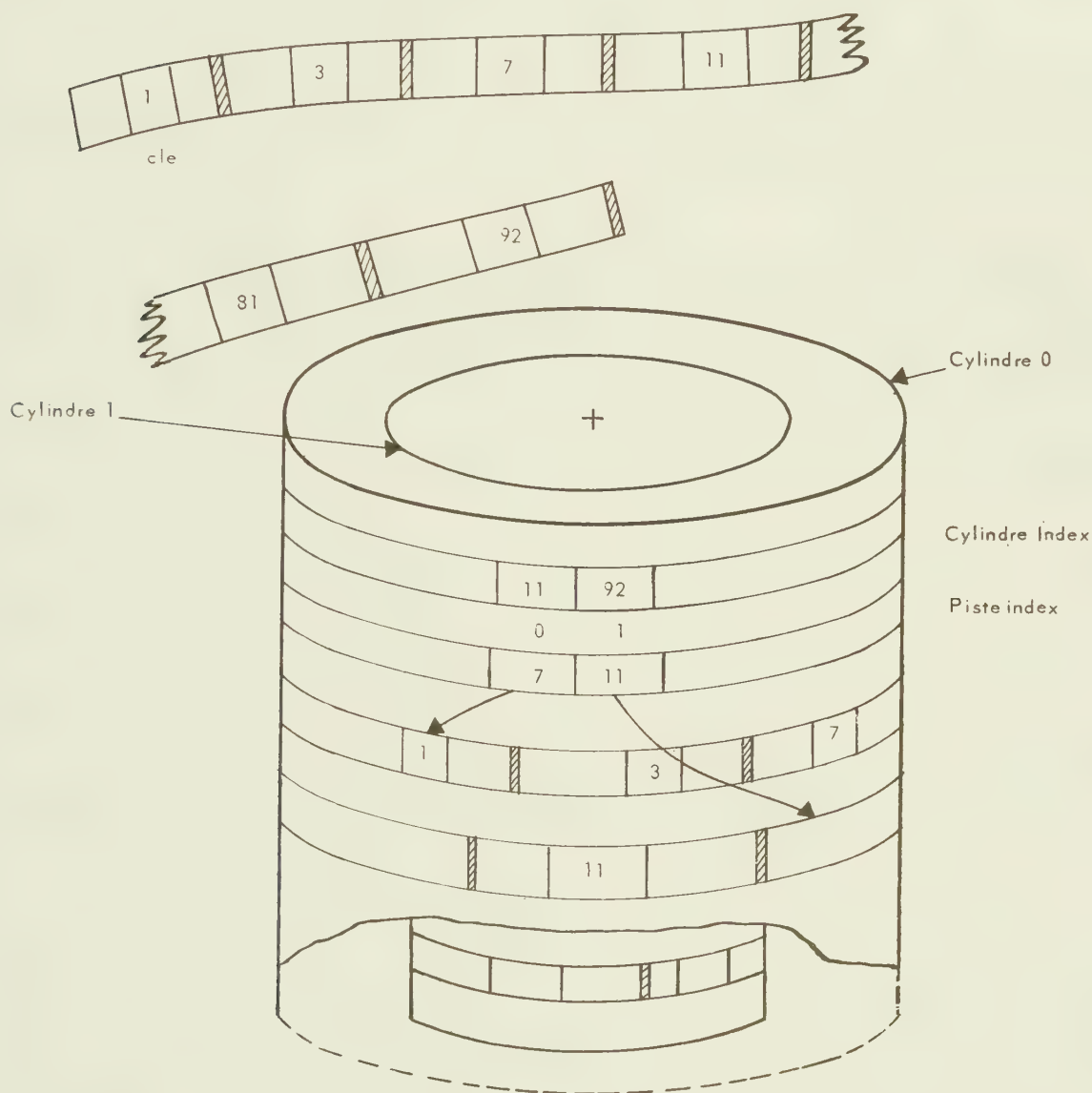
Cette organisation permet d'insérer des articles sans réécrire le fichier en totalité.

Soit par exemple, à insérer un article dont la clé vaut 599. Supposons qu'une piste est remplie par deux articles; le nouvel article doit se trouver sur la piste entre les articles de valeur de clés 598 et 600. Du fait que la piste est déjà complète, le nouvel enregistrement est écrit après celui dont la clé vaut 598 et celui dont la clé vaut 600 est réécrit sur une piste de débordement (overflow-track). La piste-index est alors modifiée afin d'indiquer qu'une piste de débordement existe.

(Lorsqu'il n'y a pas de piste de débordement disponible, l'article n'est pas écrit ; l'utilisateur en est averti).

Les deux méthodes d'accès : séquentielle et séquentielle indexée peuvent être utilisées avec cette organisation de fichier. Ainsi :

Articles triés suivant une clé :



2.9.2.3 - ORGANISATION DIRECTE

Cette organisation est exclusivement réservée aux supports appartenant à la classe STOCK ; la distinction étant faite par le mot :

STOCKDIR

Il s'agit, en général, de disques magnétiques.

Avec cette organisation pour lire ou écrire un article il faut donner au système :

- soit une adresse réelle composée des numéros du périphérique, du cylindre, de la piste et aussi de la position de l'article).
- soit une adresse relative par rapport au début du fichier.

Si chacun des articles contient une clé, il est possible de faire rechercher au système l'article qui correspond à une valeur de la clé en lui fournissant cette valeur et une adresse (réelle ou relative) à partir de laquelle commencera la recherche.

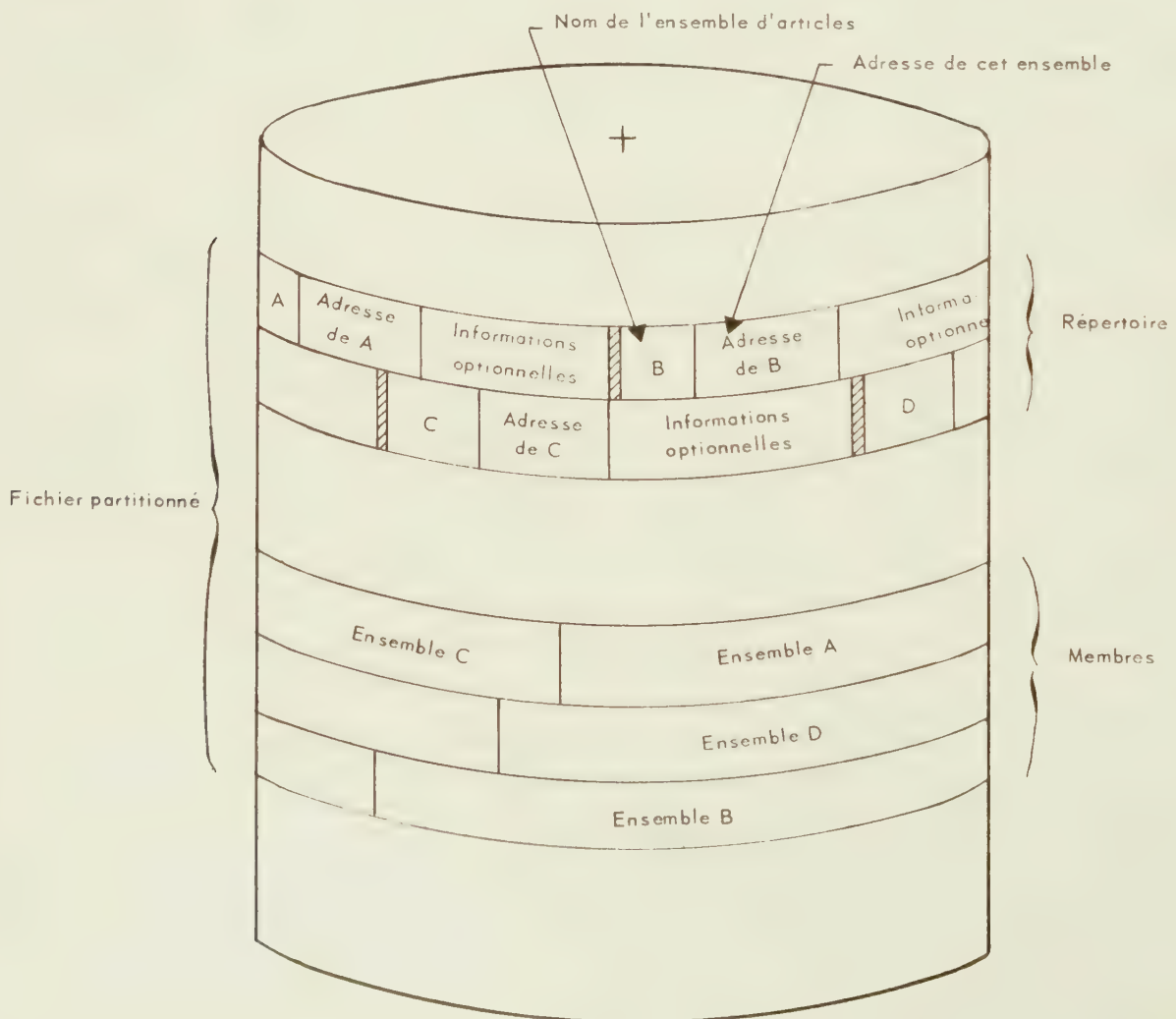
2.9.2.4 - ORGANISATION PARTITIONNEE

Cette structure possède à la fois les caractéristiques des organisations séquentielle et séquentielle indexée. Elle est exclusivement réservée aux supports appartenant à la classe **STØCK**; la distinction étant faite par le mot :

STØCKPART

Un fichier partitionné est constitué de groupes indépendants d'articles organisés séquentiellement, chacun de ces groupes étant identifié par un nom de membre dans le répertoire. Il est possible de considérer un fichier partitionné comme une bibliothèque. Chaque membre est formé d'articles organisés séquentiellement et est identifié par un nom (ou plusieurs aliases).

Le répertoire est une liste de noms de membres et il contient l'adresse de début de chaque membre. e.g.



2.9.3 - Techniques d'accès

Le système d'exploitation permet d'utiliser, essentiellement, deux techniques pour les opérations de changement de support, techniques qui sont désignées, dans la terminologie de l'Operating System IBM 360, sous les vocables de :

- Queued Access Technique - (Technique d'accès assistée),
- Basic Access Technique - (Technique d'accès non assistée)

2.9.3.1 - QUEUED ACCESS TECHNIQUE - Technique d'accès assistée

Cette technique traite les articles. Un article est introduit en mémoire par la macro-instruction GET et écrit sur un support autre que MEMOIRE par la macro-instruction PUT.

La première fois que GET est exécutée, le système introduit un bloc d'un ou plusieurs articles dans le buffer d'entrée.

A chaque nouvelle macro-instruction GET exécutée, le système fournit un nouvel article. Quand un buffer d'entrée est vide, il est automatiquement rempli par un nouveau bloc.

D'une façon similaire, la macro-instruction PUT place les articles dans le buffer de sortie et lorsqu'un buffer est plein, le système vide celui-ci sur le support de sortie.

L'utilisation de cette technique d'accès libère l'utilisateur de la plupart des préoccupations liées aux opérations d'entrée-sortie.

Cette technique introduit les articles en mémoire avant la demande (GET), elle ne peut traiter que des fichiers d'organisation séquentielle ou séquentielle indexée.

2.9.3.2 - BASIC ACCESS TECHNIQUE - Technique d'accès non assistée

Cette technique met à la disposition de l'utilisateur les macro-instructions READ et WRITE pour l'entrée et la sortie, respectivement. Elle concerne les blocs (groupement d'articles) et non les articles. En réponse à une macro-instruction READ, le système place un bloc dans le buffer d'entrée.

Si le bloc contient plusieurs articles, l'utilisateur doit se charger de les isoler les uns des autres (i.e. cette opération revient à lire une portion de file).

D'une façon similaire, la macro-instruction WRITE transmet un bloc à un fichier sur un support autre que MEMØIRE

Contrairement à la "Queued Access Technique", cette technique n'assure pas la synchronisation automatique du traitement et des opérations d'entrée-sortie. A l'exécution d'une macro-instruction READ, l'utilisateur ne peut supposer que l'article est en mémoire, comme c'est le cas avec la macro-instruction GET. Il doit tester la "complétion" de l'opération (macro-instruction CHECK) ou attendre la fin de l'opération (macro-instruction WAIT)

On reviendra plus loin sur les macro-instructions utilisées dans chacune de ces techniques

2.9.4 - Identification des fichiers

Tout fichier doit être identifié par un nom, nom qu'il faut fournir au système d'exploitation quand on veut accéder à ce fichier.

Tous les fichiers "reconnus par noms" (i.e. référencés sans indication du volume) et tous les fichiers résidents sur le même volume doivent avoir des noms distincts de tout autre fichier

Pour faciliter la chose, le système d'exploitation permet d'emploi d'expressions qualifiées, construites autrement que celles du langage : les niveaux sont inversés et les noms qui les désignent sont séparés non plus par des DE mais par des points ' '

En conséquence, un nom de fichier est un identificateur ou une suite d'identificateurs séparés par le caractère ' '; chacun d'eux doit désigner un niveau de qualification. L'ordre des niveaux est inversé par rapport aux expressions qualifiées du langage ; en d'autres termes, de gauche à droite chaque identificateur désigne une catégorie dans laquelle l'identificateur suivant désigne une sous-catégorie. Le nombre de caractères de ces identificateurs est limité à 8. Le nombre total de caractères d'un nom de fichier ne peut excéder 44, de la sorte un maximum de 22 niveaux de qualification est possible.

Pour éviter des recompilations successives, un fichier n'est pas désigné dans un programme par son nom - tel qu'on vient de le définir - mais par un renvoi. Quand le programme est

exécuté, le nom de fichier et les informations complémentaires sont spécifiées par un ordre du langage de contrôle (JCL, Job Control Language) : **Data Définition (DD) Statement**.

Dans le programme, pour accéder au fichier, on fait référence à un **Data Control Block (DCB)** associé au nom de fichier. Le DCB réalise l'interface entre le programme et l'Operating System

Le système IBM 360 permettant de disposer de nombreux périphériques, il importe de définir ce qu'est un "volume" sur chacun d'eux :

BANDE		une bobine	
STOCK	}	un dispack	
	}	un tambour	

Chaque fichier stocké sur un volume a son nom, son emplacement, son organisation, est... conservé dans le label du fichier (Data Set Label) ou dans une table : **Volume Table of Contents** uniquement pour les volumes en accès direct.

Le système permet de cataloguer automatiquement les noms de fichiers. Un fichier catalogué (ou plus exactement ayant son nom catalogué) peut être retrouvé par le système seulement par le nom. Si ce dernier est qualifié, chaque qualificateur correspond à un des index du catalogue, la recherche est effectuée en considérant l'expression de gauche à droite, i.e. du plus important au moins important.

Le système permet aussi l'affectation de numéros de génération sur les fichiers de même nom.

2.9.4.1 - VOLUMES EN ACCES DIRECT

Ce type de volume est fréquemment utilisé, car un tel volume peut contenir plusieurs fichiers et son espace peut être géré dynamiquement.

Une table, **Volume Table of Contents (VTØC)** est utilisée pour la gestion de chaque fichier et de l'espace disponible sur le volume. Chaque volume en accès direct est identifié par un label.

La VTØC décrit le contenu du volume; c'est un fichier composé d'une suite de Data Set Control Blocks (DSCB), formés d'un ou plusieurs blocs de contrôle

La VTØC peut contenir les DSCB suivants :

- un DSCB pour chaque fichier sur le volume;
- un DSCB indiquant l'espace alloué à la VTØC elle-même;
- un DSCB pour

Quand on veut stocker un fichier sur un tel volume, il faut communiquer au système la taille nécessaire au fichier. Cette taille peut être exprimée en termes de blocs, pistes ou cylindres.

2.9.4.2 - BANDES MAGNETIQUES

Du fait de l'organisation séquentielle des fichiers sur bande magnétique, le système ne permet pas les mêmes facilités que pour les volumes en accès direct. Quand un fichier est placé sur un tel volume, il faut préciser le numéro de séquence si ce fichier n'est pas le premier sur la bobine. Un volume avec un label standard ou sans label est positionné par le système de façon que le fichier puisse être lu ou écrit. Dans le cas de labels non standards, il est nécessaire de prévoir des programmes spéciaux permettant la lecture de ces labels et le positionnement.

Tous les fichiers sur un même volume doivent être créés avec la même densité d'enregistrement.

2.9.5 - Formats des articles des fichiers

L'article est l'unité de base d'information pour le traitement. Avant d'écrire les articles sur un volume, il est possible de les grouper : on dit alors qu'il sont bloqués.

Un bloc est composé de données figurant entre deux "intervalles entre enregistrements" (Intereccord Gap-IRG). Chaque bloc peut être constitué d'un ou de plusieurs articles groupés.

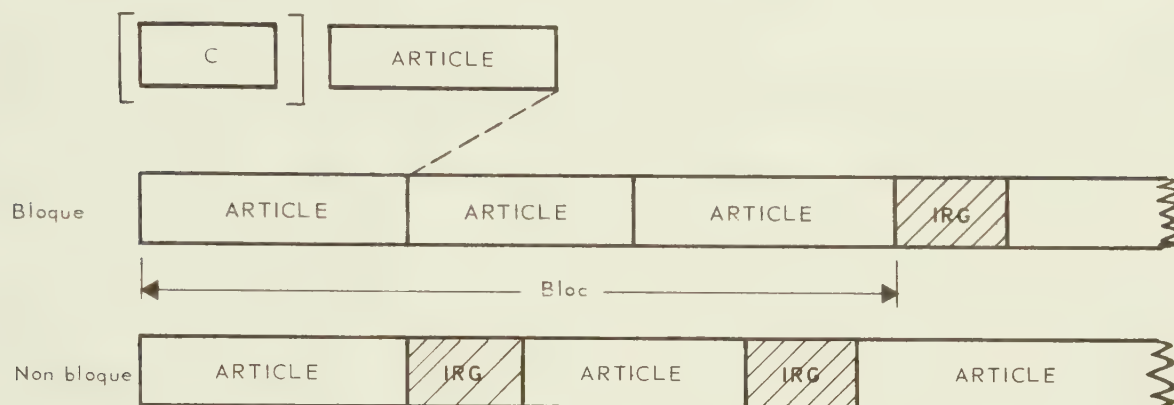
La technique consistant à bloquer les articles permet d'économiser la place occupée sur le volume et le temps de traitement du fichier.

Les enregistrements sont de l'un des trois formats suivants :

- longueur constante (F)

- longueur variable (V)
- longueur indéfinie (U)

2.9.5.1 - ARTICLES DE LONGUEUR CONSTANTE



La taille des articles est constante dans tout le fichier. Dans le cas où les articles sont bloqués le nombre d'articles dans le bloc est généralement constant. Dans le cas contraire chaque article constitue un bloc

Les articles de format F (longueur constante) sont plus faciles à traiter, pour le système, que les articles de format V (longueur variable). Un fichier séquentiel est dit contenir des articles dans la forme standard F, si :

- tous les articles du fichier sont de format F;
- chaque piste (sur un volume en accès direct) exceptée la dernière est complète;
- aucun bloc, à part le dernier, n'est tronqué.

Le caractère C (1 octet) est facultatif, il ne sert que pour les opérations sur imprimante ou perforateur par exemple.

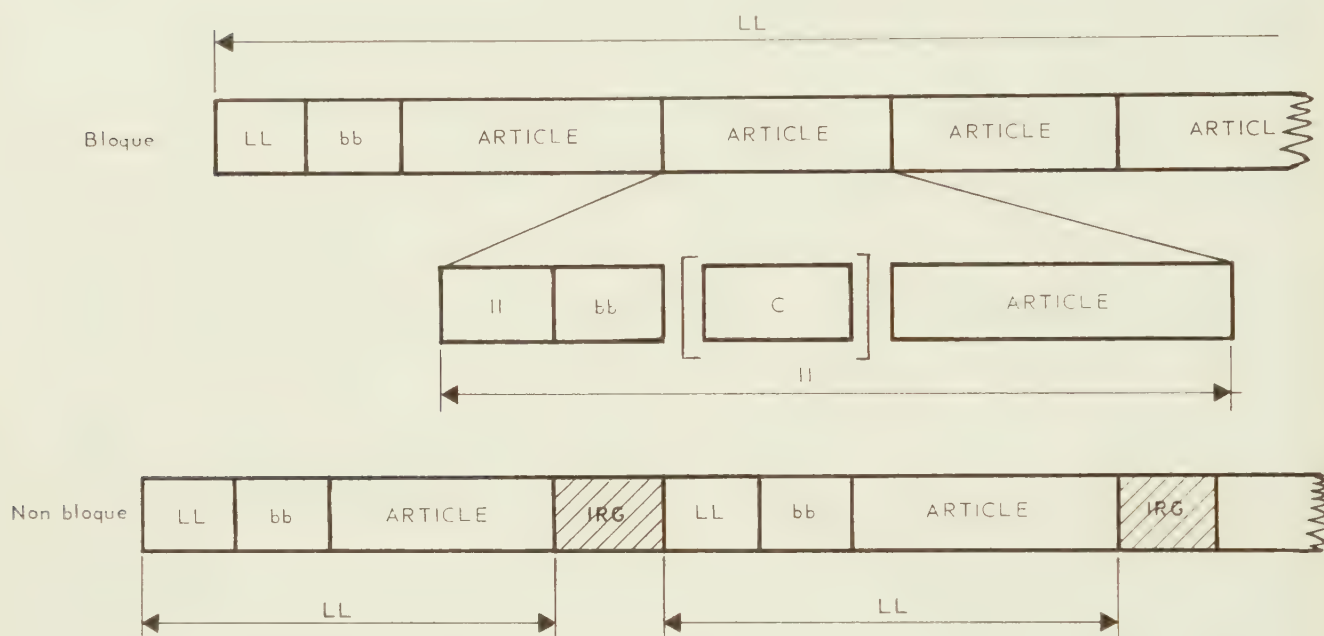
2.9.5.2 - ARTICLES DE LONGUEUR VARIABLE

Dans ce cas les articles et les blocs sont de longueur variable, chacun d'eux spécifiant sa propre longueur. Comme il est indiqué sur la figure ci-dessous, les quatre premiers octets d'un tel article contiennent des informations de contrôle :

- 2 octets-longueur de l'article (ll)
- 2 octets à l'usage du système (bb)

Le système effectue automatiquement les tests de longueur et le bloquage-débloquage.

Le caractère facultatif C (contrôle) est le 5ème octet de l'enregistrement lorsqu'il est présent.



La longueur du bloc LL est représentée sur 2 octets : 'bb' est un demi-mot (2 octets) réservé à l'usage du système.

Quand les articles sont bloqués par le système, ces caractères sont automatiquement calculés, dans le cas contraire l'utilisateur doit réaliser lui-même cette opération

Ces 8 ou 9 caractères de contrôle du bloc ne sont pas imprimés ou perforés

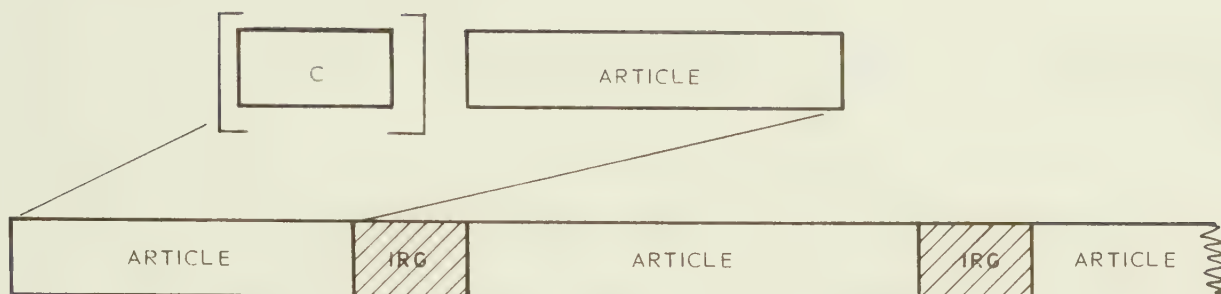
2.9.5.3 - ARTICLES DE LONGUEUR INDEFINIE

Le format U permet de traiter des articles qui ne sont ni de format F ni de format V. Chaque bloc est traité comme un article, l'utilisateur doit faire le déblocage.

Du fait que le système ne fait pas la vérification de longueur sur les articles de format U, l'utilisateur doit lire au moins un bloc complet.

2.9.5.4 - CARACTERE DE CONTROLE

Il est possible de spécifier, dans l'ordre DD, dans la macro-instruction DCB (2.9.6), ou dans le label du fichier qu'un caractère de contrôle fait partie de chaque article. Ce caractère



est utilisé par exemple, pour l'imprimante ou comme sélecteur de case sur un perforateur de cartes. Ce caractère n'est pris en compte que par les périphériques qui l'utilisent, les autres le considèrent au même titre que le reste de l'article.

2.9.6 - Interface avec le système d'exploitation

Les informations décrivant un fichier et son support sont placées dans le **Data Control Block (DCB)**. Le DCB est créé dans un programme par la macro-instruction DCB.

Cependant les informations contenues dans le DCB proviennent de sources différentes :

- de la macro-instruction DCB ;
- de l'ordre du langage de contrôle (JCL) DD ;
- du label du fichier

La macro-instruction **OPEN** d'ouverture du fichier fournit au DCB des informations supplémentaires. Il est aussi possible de modifier certaines zones du DCB en cours d'exécution.

Les différentes informations contenues dans un DCB sont :

1 - Organisation du fichier- Data Set Organization (DSORG)
spécifie l'organisation du fichier :

Séquentiel physique	PS
Séquentiel indexé	IS

Partitionné	PØ
Direct	DA

2 - Format de l'article - Record format (RECFM)

décrit le format de l'article :

Longueur fixe	F
Longueur variable	V
Longueur indéfinie	U

Le blocage est indiqué par la lettre B, i.e. FB et VB. Le débordement de piste est spécifié par la lettre T (Track overflow), i.e. FBT, VBT.

3 - Longueur de l'article - Record Length (LRECL)

spécifie la longueur en octets des articles dans un fichier. Pour le format V, il s'agit de longueur maximale. Ce renseignement peut être omis pour le format U.

4 - Taille du bloc - Block Size (BLKSIZE)

spécifie la taille maximale en octets d'un bloc. Si les articles sont de format F, cette taille doit être un multiple de la longueur de l'article (LRECL)

Si les articles sont de format V, la taille du bloc indiquée doit être la taille maximale et si les articles ne sont pas bloqués, la taille du bloc est la longueur de l'article (LRECL) augmentée de 4.

Les paramètres suivants de l'ordre DD, servent à compléter le DCB; ils sont communs à la plupart des organisations et des périphériques :

1 - Data Definition Name (DDNAME)

est le nom de l'ordre DD, il permet une relation logique avec le DCB associé qui portera le même nom (DDNAME).

2 - Nom de fichier. Data Set Name (DSNAME)

spécifie le nom du fichier (qu'il soit nouveau ou déjà existant).

3 - Data Control Block (DCB)

permet à l'aide de paramètres de compléter le DCB, mais en aucun cas de le modifier.

4 - Spécification de canal - Channel Separation and Affinity (SEP/AFF)

indique que certains fichiers doivent utiliser des canaux différents durant les opérations d'entrée-sortie.

5 - Périphérique - Input-output Device (UNIT)

spécifie le nombre et le type de périphériques à allouer au fichier

6 - Occupation STOCK - Space Allocation (SPACE)

désigne la place à allouer sur un volume du type accès-direct pour un fichier. L'espace inutilisé est restitué, au système, à la fin du traitement

7 - Identification du volume - Volume Identification (VOLUME)

identifie le ou les volumes particuliers, ou spécifie le nombre de volumes à assigner à un fichier ou encore les volumes contenant le fichier.

8 - Label du fichier - Data Set Label (LABEL)

indique le type et le contenu du ou des labels associés au fichier. Le système vérifie les labels standards (SL) ou les labels utilisateur standards (SUL). Les labels non standards (NSL) sont traités par des programmes spéciaux de l'utilisateur

9 - Data Set Disposition (DISP)

décrit le statut du fichier et indique ce qui doit en être fait à la fin du traitement.

Les paramètres suivants sont relatifs aux macro-instructions utilisées. Ils décrivent le traitement des fichiers par le programme.

1 - Macro-Instruction Form (MACRF)

spécifie non seulement les macro-instructions utilisées dans le programme mais aussi le mode de traitement. L'organisation du fichier, la forme des macro-instructions et le mode de traitement déterminent les programmes que le système doit employer durant l'exécution

2 - Sorties spéciales - Exits to Special Processing Routines

La macro-instruction DCB peut être utilisée pour identifier :

- un programme traitant les fins de fichiers;
- un programme remplaçant le système pour la reprise des erreurs ;

ou une liste contenant les adresses de ces programmes.

Les différentes possibilités sont :

2.1 - Fin de fichier End-of-Data-Set-Exit-Routine (EODAD)

indique un programme prenant en compte le traitement après qu'un ordre de lecture (GET ou READ) ait détecté la fin du fichier. Ce programme doit aussi prendre en compte certaines conditions liées à l'utilisation des macro-instructions READ et CHECK.

En cas d'absence de cette indication, l'exécution est suspendue.

2.2 - Erreurs d'entrée-sortie - Synchronous Error Routine Exit (SYNAD)

indique un programme prenant en compte les erreurs d'entrée-sortie. Le système fournit des indications à ce programme sur la nature de l'erreur.

En l'absence de cette indication l'exécution est suspendue

2.3 - Procédures spéciales Exit List (EXLST)

indique un programme ou une série de programmes prenant en compte certains travaux particuliers :

1. Traitement d'un label utilisateur de tête en entrée.
2. Traitement d'un label utilisateur de tête en sortie
3. Traitement d'un label utilisateur de fin en entrée
4. Traitement d'un label utilisateur de fin en sortie.
5. Programme permettant de compléter le DCB lors de l'ouverture du fichier (OPEN).

2.9.7 - Techniques de traitement des données

Une méthode d'accès peut être définie comme étant la combinaison de l'organisation du fichier et de la technique utilisée pour traiter ces données.

Il existe deux techniques d'accès précédemment mentionnées (2.9.3). Nous allons les examiner successivement.

2.9.7.1 - QUEUED ACCESS TECHNIQUE - Technique d'accès assisté

Les macro-instructions GET et PUT sont relatives à cette technique. Elles provoquent automatiquement le blocage-déblocage des articles lus ou écrits.

Le système synchronise les opérations d'échange et le traitement.

Les branchements aux programmes d'erreurs utilisateur (SYNAD et EODAD) sont réalisés automatiquement, si nécessaire.

2.9.7.1.1 - GET - OBTENTION D'UN ARTICLE

La macro-instruction GET permet d'obtenir un article depuis un fichier "entrée". Elle opère d'une façon séquentielle indépendamment du périphérique. Au besoin, GET gère les buffers d'entrée, déblocage les articles, passe le contrôle aux programmes utilisateur de

traitement d'erreur. Lorsque GET détecte une marque de fin de fichier, le système teste automatiquement le label de fin et passe le contrôle au programme utilisateur qui traite la fin de fichier (EØDAD).

Lorsque le système rencontre une marque de fin de volume, il change automatiquement de volume si le fichier s'étend sur plusieurs volumes ou s'il s'agit de fichiers carcaténés.

2.9.7.1.2 - PUT - ECRITURE D'UN ARTICLE

La macro-instruction PUT place un article dans un fichier "sortie". Comme GET, elle opère d'une façon strictement séquentielle et indépendamment du support. Au besoin, PUT gère les buffers, bloque les articles, passe le contrôle au programme de traitement d'erreur. Elle recherche les volumes et crée automatiquement les labels.

Si PUT est employée pour un perforateur de cartes ou une imprimante, le système ajuste le nombre d'articles par bloc (Format V ou F) à 1.

2.9.7.1.3 - PUTX - ECRITURE D'UN ARTICLE MIS A JOUR

La macro-instruction PUTX est utilisée pour mettre à jour un fichier ou pour créer un fichier en utilisant comme base, les articles d'un fichier "entrée".

2.9.7.2 - BASIC ACCESS TECHNIQUE - Technique d'accès non-assisté

Elle est utilisée quand le système ne peut prévoir l'ordre dans lequel les articles seront traités ou lorsque l'utilisateur ne désire pas tout ou partie des traitements automatiques réalisés par la "Queued Access Technique". Dans ce cas le système ne gère pas les buffers et ne synchronise pas les opérations d'entrée-sortie avec le traitement, mais des macro-instructions permettent de réaliser ces fonctions.

Les macro-instructions READ et WRITE sont relatives à cette technique.

READ et WRITE traitent les blocs et non les articles ; l'utilisateur doit donc bloquer et débloquer ces articles.

READ et WRITE initialisent l'opération d'échange, l'utilisateur peut poursuivre le traitement, mais avant de traiter l'information reçue il doit vérifier que l'opération est terminée. Pour cela il doit recourir à la macro-instruction CHECK ; il peut aussi attendre que l'opération soit terminée à l'aide de la macro-instruction WAIT.

Le nombre d'instructions READ et WRITE qu'il est possible de faire simultanément - c'est-à-dire avant la première macro-instruction CHECK - est indiqué par le paramètre NCP (Nombre de Programmes Canal).

2.9.7.2.1 - READ - LECTURE D'UN BLOC

READ recherche un bloc dans un fichier d'entrée et le place en mémoire. Le contrôle est redonné à l'utilisateur après le lancement de l'opération. L'utilisateur doit alors s'assurer que l'opération est terminée en se référant au **Data Event Control Block (DECB)** créé pour l'opération. Ce DECB ne peut être réutilisé avant que l'opération ne soit terminée

READ nécessite un paramètre supplémentaire pour préciser l'organisation du fichier et le traitement que doit réaliser le système ; ce paramètre est choisi dans l'une des trois classes :

1 - Séquentiel

SF - lire le fichier séquentiellement ;

SB - lecture arrière du fichier, uniquement sur BANDE en format F ou U.

2 - Séquentiel indexé

K - lire le fichier ;

KU - lecture pour mise à jour, le système conserve l'adresse de l'enregistrement sur le périphérique et lorsque WRITE réécrit cet enregistrement ; la recherche d'index n'est pas nécessaire pour le système.

3 - Direct

D - utilisation de la méthode d'accès direct ;

I - recherche de l'adresse du bloc en utilisant son identification,

K - recherche de l'adresse du bloc en utilisant une clé ;

F - conservation de l'adresse de l'article (feedback) ;

X - maintient du contrôle exclusif du bloc

2.9.7.2.2 - WRITE - ECRITURE D'UN BLOC

WRITE place un bloc, situé en mémoire, dans un fichier de sortie. WRITE peut être utilisé pour la mise à jour d'un fichier. Les mêmes considérations que pour READ s'appliquent à WRITE (DECB).

Les paramètres supplémentaires sont :

1 - Séquentiel

SF - écrire le fichier séquentiellement ;

2 - Séquentiel indexé

K - écrire un bloc contenant un article mis à jour, ou remplacer un article par un autre non bloqué ayant la même clé. L'article à remplacer n'a pas besoin d'avoir été introduit en mémoire auparavant.

KN - écrire un nouvel article.

3 - Direct

SD - écrire un article sans signification (dummy) de longueur fixe ;

- SZ - écrire un article dit de "capacité": le système en fournit le contenu, écrit cet article et "avance" à la piste suivante;
- D - utiliser la méthode d'accès direct ;
- I - rechercher si l'argument identifie un bloc;
- K - rechercher si l'argument est une clé;
- A - ajouter un nouveau bloc;
- F - donner l'adresse d'un article (feedback);
- X - supprimer le contrôle exclusif de l'article.

2.9.7.2.3 - CHECK - TEST DE LA FIN D'UNE OPERATION READ WRITE

Dans le traitement d'un fichier du type séquentiel ou accès direct, l'utilisateur peut tester et attendre la fin d'une opération READ ou WRITE grâce à la macro-instruction CHECK.

CHECK passe le contrôle aux programmes spécifiques pour les traitements particuliers (SYNAD ou EØDAD). Il réalise aussi, automatiquement, les procédures de fin de volume, à savoir :

2.9.7.2.4 - WAIT - ATTENTE DE LA FIN D'UNE OPERATION READ/WRITE

Dans le traitement d'un fichier d'organisation séquentielle indexée, l'utilisateur doit tester la fin des opérations à l'aide de la macro-instruction WAIT.

Dans ce cas le test des erreurs, des fins de fichier n'est pas automatique : il est à la charge de l'utilisateur.

WAIT peut être utilisée pour attendre la fin d'une série de READ/WRITE, mais chacune des opérations de cette série doit être testée séparément.

2.9.7.2.5 - DATA EVENT CONTROL BLOCK (DECB)

Un DECB est une zone mémoire qui est analysée par le système à la fin des opérations d'entrée-sortie pour déceler d'éventuelles erreurs ; s'il y en a, il passe le contrôle au programme spécifique de l'utilisateur (SYNAD) ; si un tel programme n'a pas été fourni l'exécution est suspendue, sauf si le traitement concerne un fichier d'organisation séquentielle indexée

Dans ce cas en effet, il n'est pas possible à l'utilisateur de prévoir des programmes de traitement d'erreur (SYNAD) du fait qu'il ne lui est pas possible d'utiliser CHECK. En conséquence, l'utilisateur doit analyser lui-même le contenu du DECB

2.9.8 - Traitement des erreurs

2.9.8.1 - SYNADAF - ANALYSE DES CONDITIONS SYNAD

La macro-instruction SYNADAF permet d'analyser les conditions contenues dans le DECB et produit un message significatif que l'utilisateur peut insérer dans un fichier approprié.

Il est judicieux de l'employer au début du programme de traitement d'erreur.

2.9.8.2 - SYNADRLS - RETOUR D'UN PROGRAMME SYNAD

Cette macro-instruction SYNADRLS doit être utilisée à la fin d'un programme SYNAD d'analyse d'erreur.

2.9.9 - Choix d'une méthode d'accès

Le choix d'une méthode d'accès est lié en premier lieu à l'organisation du fichier

Le tableau suivant résume les possibilités :

Organisation du fichier	Technique d'accès	
	Basic	Queued
Séquentiel	BSAM	QSAM
Partitionné	BPAM	
Séquentiel - indexé	BISAM	QISAM
Direct	BDAM	
Télécommunications	BTAM	QTAM

(Le traitement pour les télécommunications n'est pas explicité ici.)

L'utilisateur peut aussi contrôler directement les périphériques, tant au point de vue du traitement que de l'organisation, à l'aide de la macro-instruction EXCP-Execute channel Program.

2.9.10 - Ouverture et fermeture des fichiers

Avant de faire quelque opération que ce soit sur un fichier il est nécessaire de l' "ouvrir" à l'aide de la macro-instruction **OPEN** ; de même, après traitement, il est nécessaire de le "fermer" à l'aide de la macro-instruction **CLOSE**. Ces deux macro-instructions permettent au système de faire automatiquement une série d opérations.

2.9.10.1 - OPEN - OUVERTURE D'UN FICHIER

OPEN est utilisée pour compléter le DCB associé au fichier.

1 - Méthode de traitement

Un fichier peut être traité soit comme entrée (**DONNEE**) ou comme sortie (**RESULTAT**) (**INPUT** ou **OUTPUT**) ou comme une combinaison des deux (**MIXTE**) (**INPUT**, **OUTPUT** - en BSAM seulement).

Si le fichier est sur un volume en accès direct, les enregistrements peuvent être mis à jour (**UPDAT**).

Les volumes bande magnétique peuvent être traités en lecture arrière (**RDBACK** - en BSAM et QSAM seulement).

2 - Positionnement du volume

Quand une marque de fin de fichier est détectée, le système positionne le volume d'après les paramètres de l'ordre **DD** à moins que **OPEN** n ait fourni les informations nécessaires. Les paramètres pour les fichiers séquentiels sont :

LEAVE - positionne le volume à la fin logique du fichier;

REREAD - positionne le volume au début logique du fichier.

Ces paramètres ne peuvent être utilisés que pour les périphériques des types : **BANDE** et **STOCK**.

Le positionnement des bandes magnétiques est fonction de la dernière opération de lecture et de l'existence de labels.

2.9.10.2 - CLOSE - FERMETURE D'UN FICHIER

La macro-instruction **CLOSE** est utilisée pour clore le traitement d'un fichier. Il est possible de spécifier le repositionnement du fichier à la fermeture.

Le système permet de fermer temporairement les fichiers traités en BSAM. Avec cette option la macro-instruction **CLOSE** permet d'effectuer le traitement des labels et le repositionnement,

mais le fichier est toujours considéré comme étant ouvert, ce qui évite d'utiliser une macro-instruction **OPEN** avant de traiter le fichier dans la suite du programme.

Pour les bandes magnétiques, la présence ou l'absence de labels est sans influence sur le repositionnement.

2.9.10.2.1 - TRAITEMENT DE LA FIN DE VOLUME

Le contrôle est passé automatiquement au système si l'une des conditions suivantes est détectée :

- Indicateur de fin de fichier (INPUT)
- Marque de bande (INPUT)
- Marque de file - File Mark (INPUT sur volume en accès direct)
- Fin de bobine (OUTPUT)
- Fin de zone (OUTPUT sur volume en accès direct).

Il est possible d'utiliser la macro-instruction **FEØV** (Force End-of-Volume) avant que la fin de volume soit détectée.

Le système crée ou teste les labels de fin éventuellement, ou passe le contrôle au programme utilisateur approprié. Il peut aussi assurer le changement de volume dans le cas d'un fichier multi-volume.

2.9.10.2.2 - FEØV - CREATION D'UNE FIN DE VOLUME

La macro-instruction **FEØV** permet au système d'initialiser les procédures de traitement de fin de volume avant la fin physique du volume courant. Elle ne peut être utilisée que pour le traitement séquentiel, i.e. **QSAM** et **BSAM**

2.9.11 - Acquisition et contrôle des buffers

Le système permet plusieurs méthodes pour la gestion des buffers. Chaque **buffer**, i.e. portion de mémoire utilisée pour le stockage intermédiaire des articles, correspond usuellement en longueur à la taille d'un bloc du fichier traité. Avec la "Queued access technique", toute référence à un buffer se réfère en réalité à l'article suivant, i.e. **segment de buffer**.

Il est possible d'associer plusieurs buffers à un fichier, il s'agit alors d'un pool de buffers.

Les segments de buffer et les buffers dans un pool de buffers sont gérés automatiquement par le système lorsque la "queued access technique" est utilisée. Cependant l'utilisateur peut suspendre le traitement d'un buffer en utilisant soit la macro-instruction RELSE (entrée) ou la macro-instruction TRUNC (sortie).

Deux techniques de "bufferisation" : simple ou avec échange sont utilisables pour les fichiers séquentiels. Pour les fichiers du type séquentiel - indexé, seule la bufferisation simple est utilisable.

Avec la "basic access technique", l'usager peut utiliser les buffers comme zone de travail. Il peut les contrôler directement, en utilisant les macro-instructions GETBUT et FREEBUF, ou dynamiquement en le spécifiant dans les macro-instructions DCB et READ WRITE. Dans le second cas, le système créera un buffer à chaque opération READ, buffer qui sera supprimé par les macro-instructions WRITE ou FREEDBUF.

2.9.11.1 - CONSTRUCTION D'UN POOL DE BUFFERS

La construction peut s'opérer de trois façons :

- statiquement par la macro-instruction BUILD
- explicitement par la macro-instruction GETPOOL
- automatiquement par le système à l'ouverture du fichier.

Si la macro-instruction BUILD ou le contrôle dynamique des buffers est utilisé, les buffers sont restitués au pool à la fermeture du fichier. Si le pool de buffers est construit explicitement ou automatiquement, la zone doit être restituée au système par la macro-instruction FREEPOL.

Il est possible de spécifier l'alignement (simple ou double-mot) d'un buffer.

2.9.11.1.1 - BUILD - CONSTRUCTION D'UN POOL DE BUFFERS

La macro-instruction BUILD réserve et structure une portion de mémoire. Les 8 premiers octets d'un pool de buffers constituent le Buffer Pool Control Block (BUFCB). Le nombre de buffers et leur longueur sont spécifiés par les paramètres BUFNØ et BUFL.

Cette zone peut être réutilisée après la fermeture du fichier.

Le même pool de buffers peut être utilisé pour plusieurs fichiers de caractéristiques identiques, en fait il faut construire suffisamment de buffers pour pouvoir traiter tous les fichiers.

2.9.11.1.2 - GETPØØL - GENERATION D'UN PØØL DE BUFFERS

Il est possible de laisser le nombre de buffer et leur longueur en paramètres et de ne les fixer qu'à l'exécution par la macro-instruction GETPØØL.

Elle structure une partie d'un pool de buffers, génère le BUFCB et associe le pool ainsi obtenu à un fichier particulier.

2.9.11.1.3 - CONSTRUCTION AUTOMATIQUE D'UN PØØL DE BUFFERS

En l'absence des macro-instructions précédentes, le système construit automatiquement le pool de buffers. Cependant, si l'utilisateur emploie la "basic access technique" pour traiter un fichier séquentiel-indexé ou un fichier en accès direct, il doit spécifier au système le contrôle dynamique des buffers, autrement le système ne peut construire le pool de buffers

2.9.11.1.4 - FREEPØØL - LIBERATION D'UN PØØL DE BUFFERS

Un pool de buffer assigné à un fichier soit automatiquement par la macro-instruction ØPEN (sauf dans le cas du contrôle dynamique) soit explicitement par GETPØØL doit être restitué au système avant la fin du programme. C'est le rôle de la macro-instruction FREEPØØL.

En règle générale :

- si la "queued access technique" est utilisée, un fichier de sortie doit être fermé en premier pour être assuré que tous les articles ont bien été écrits.
- dans le cas de la bufférisation avec échange ou dans le traitement d'un fichier séquentiel indexé utilisant la "queued access technique", le buffer pool ne doit pas être restitué avant que tous les fichiers n'aient été fermés.

2.9.11.2 - CONTROLE DES BUFFERS

Quatre techniques peuvent être utilisées pour contrôler les buffers :

- simple ou avec échange pour la "queued access technique";
- directe ou dynamique pour la "basic access technique".

Seule la "bufférisation" simple peut être utilisée pour traiter un fichier séquentiel indexé : les segments de buffer et les buffers, dans un pool de buffers sont alors contrôlés automatiquement par le système.

La "queued access technique" permet 3 modes de traitement en ce qui concerne le mouvement de l'information en mémoire :

- **Move mode** : l'article est déplacé depuis la zone utilisateur dans le buffer de sortie, ou depuis le buffer d'entrée dans la zone utilisateur.
- **Locate mode** : l'article n'est pas déplacé; le système garnit le registre 1 avec l'adresse du prochain buffer d'entrée ou de sortie
- **Substitute mode** : l'article n'est pas déplacé, cependant l'adresse du prochain buffer d'entrée ou de sortie est échangée avec l'adresse de la zone utilisateur.

Deux modes de traitement de la macro-instruction PUTX peuvent être utilisés en relation avec une macro-instruction GET-locate :

- le mode mise à jour (UPDATE) réécrivant l'article mis à jour dans le fichier d'où il provient.
- le mode ~~Ø~~OUTPUT transférant l'article dans un fichier de sortie.

Il n'y a pas de mouvement de l'article en mémoire. Le mode de traitement est indiqué par le paramètre MACRF de la macro-instruction DCB.

Avec la "basic access technique", il est possible de contrôler les buffers de l'une des deux façons suivantes :

- **directement** à l'aide de GETBUF et de FREEBUF.
- **dynamiquement** en le spécifiant dans la macro-instruction READ ou WRITE. C'est le seul type de contrôle que l'utilisateur puisse utiliser pour la mise à jour d'un fichier séquentiel indexé ou en accès direct ou l'addition de blocs à un tel fichier. Un buffer "dynamique" ne fait pas partie d'un pool de buffers. Il peut être restitué au système par la macro-instruction WRITE en mise à jour ; autrement il doit être restitué par la macro-instruction FREEDBUF.

2.9.11.2.1 - "BUFFERISATION SIMPLE"

Le terme "bufférisation simple" se réfère aux relations entre les segments à l'intérieur d'un buffer. Tous les segments dans un tel buffer sont contigus en mémoire et toujours associés au même fichier. Quand le système construit le pool de buffers, il crée un mot de commande canal (Channel Command Word - CCW) pour chacun des buffers du pool.

En conséquence, chaque article doit être déplacé (physiquement) d'un buffer d'entrée, dans un buffer de sortie, si le traitement est de ce type. Un article peut être traité dans le buffer.

Avec la bufférisation simple il est possible de traiter des articles de tout format.

Les articles peuvent être traités et déplacés de l'une ou l'autre des façons suivantes :

1. GET-locate, PUT-move./PUTX-output.
articles traités dans le buffer d'entrée et déplacés dans le buffer de sortie.
2. GET-move, PUT-locate.
articles déplacés d'un buffer d'entrée dans un buffer de sortie où ils peuvent être traités.
3. GET-move, PUT-move.
articles déplacés d'un buffer d'entrée dans une zone de travail, traités, déplacés dans un buffer de sortie
4. GET-locate, PUTX-update.
articles traités dans le buffer d'entrée et réécrits dans le fichier.

2.9.11.2.2 - "BUFFERISATION" AVEC ECHANGE

Comme précédemment, ce terme se réfère aux relations entre segments dans un buffer. Dans un tel buffer, tous les segments ne sont pas nécessairement contigus ; mais ils sont toujours associés au même fichier. Quand le pool de buffers est construit, le système crée un CCW pour chaque segment de buffer, et de ce fait il est possible d'échanger les CCW. Pour utiliser la "bufférisation" avec échange, l'usager doit prévoir une zone de travail de même taille et de même alignement qu'un segment de buffer. Cette zone de travail peut alors être échangée avec un segment de buffer ; en d'autres termes, le CCW créé pour le segment de buffer peut être associé à la zone de travail.

Cette technique permet d'éviter les mouvements d'information. Elle ne présente d'avantage que si le "substitute mode" ou le mode PUTX - output est utilisé.

Dans l'utilisation de cette technique, les conditions suivantes doivent être satisfaites :

- les buffers d'entrée et de sortie doivent être de même taille et de même alignement ;
- les articles ne doivent pas être bloqués, ou, s'il le sont, bloqués uniquement en format F ;
- le débordement de piste (Track overflow) ne peut pas être utilisé avec les articles bloqués (de format F) ;
- les modes GET-move et PUT locate ne peuvent pas être utilisés

Les articles peuvent être traités de l'une ou l'autre des façons suivantes :

1. GET-substitute, PUT-substitute.

2. GET-locate, PUTX-output.

3. GET-locate, PUT-substitute.

Dans ce dernier cas le programme de l'utilisateur doit déplacer l'information

Le tableau suivant résume les techniques de bufférisation et les modes de traitement que l'utilisateur peut utiliser

Fichier de sortie Fichier d'entrée		Bufférisation simple		Bufférisation avec échange	
		PUT-locate	PUT-move	PUT-move	PUT-substitute
Bufférisation simple	GET-move	*	*	*	*
	GET-locate	*	*	*	*
Bufférisation avec échange	GET-locate	*	*	*	*
	GET-substitute	*	*	*	*
Le programme doit déplacer l'article		* * *			* *
Le système déplace l'article		*	* * * *	* * * *	*
L'article n'est pas déplacé					*
Zone de travail nécessaire		*	* *	* *	* * * *
PUTX-output peut être utilisée			* *	* *	

2.9.11.3 - RELSE - ABANDON D'UN BUFFER D'ENTREE

Lorsque l'utilisateur utilise la "queued access technique" pour traiter un fichier séquentiel indexé ; il peut, en utilisant la macro-instruction RELSE, indiquer au système qu'il désire ignorer les articles suivants encore disponibles dans le buffer courant. En réponse à la prochaine macro-instruction GET, le système délivrera un article depuis un autre buffer.

2.9.11.4 - TRUNC - TRONCATION D'UN BUFFER DE SORTIE

Lorsque l'utilisateur utilise la "queued access technique" pour traiter un fichier séquentiel ; il peut, en utilisant la macro-instruction TRUNC, demander au système d'écrire un bloc court.

2.9.11.5 - GETBUT - OBTENTION D'UN BUFFER APPARTENANT A UN POOL

Lorsque l'utilisateur utilise la "basic access technique" ; il peut, à l'aide de la macro-instruction GETBUF, demander au système de lui fournir l'adresse d'un buffer dans un pool construit par l'une des macro-instructions suivantes :

BUILD

GETPØØL

ØPEN

2.9.11.6 - FREEBUF - RESTITUTION D'UN BUFFER A UN POOL

Lorsque l'utilisateur utilise la "basic access technique" et lorsqu'il a obtenu un buffer appartenant à un pool à l'aide de la macro-instruction GETBUF alors il peut restituer le buffer à ce pool grâce à la macro-instruction FREEBUF.

2.9.11.7 - FREEDBUF - RESTITUTION D'UN BUFFER "DYNAMIQUE" A UN POOL

Lorsque l'utilisateur utilise l'option de "bufférisation dynamique" il doit restituer, au système, le buffer ainsi obtenu à l'aide de la macro-instruction FREEDBUF

Le tableau suivant résume les possibilités pour les différentes méthodes d'accès.

Organisation	Séquentiel		Partitionné	Séquentiel indexé			Direct
Méthode d'accès	QSAM	BSAM	BPAM	QISAM		BISAM	BDAM
				LØAD	SCAN		
Macro-instructions de base	GET, PUT, PUTX	READ, WRITE	READ, WRITE, FIND, STØW	PUT	SETL, GET, PUTX	READ, WRITE	READ, WRITE
Synchronisation	Automatique	CHECK	CHECK	Automatique	Automatique	WAIT	WAIT
Format de l'article	Logique F, V Bloc U	Bloc F, V, U	Bloc F, U, V	Logique F, V	Logique F, V	Logique F, V	Bloc F, V, U
Création des buffers et construction	BUILD GETPØØL Automatique	BUILD GETPØØL Automatique	BUILD GETPØØL Automatique	BUILD GETPØØL Automatique	BUILD GETPØØL Automatique	BUILD GETPØØL Automatique	BUILD GETPØØL Automatique
Technique de bufférisation	Automatique Simple Exchange	GETBUF FREEBUF	GETBUF FREEBUF	Automatique Simple	Automatique Simple	GETBUF FREEBUF Dynamique FREEDBUF	GETBUF FREEBUF Dynamique FREEDBUF
Modes	Move Locate Substitute			Move Locate	Move Locate		

2.9.12 - Adaptation au langage

Toutes les possibilités précédentes sont, en général, réservées aux usagers program-mand en assembleur. Nous les avons mises à la disposition des usagers qui utiliseraient le lan-gage en les y incorporant. Cependant il n'est pas encore possible de décrire avec précision tant les formes particulières que les conditions se rapportant à chaque macro-instruction. Toute une série d'essais seront nécessaires pour définir cette partie et nous n'allons qu'évoquer les différentes macro-instructions.

Dans le langage, deux classes de supports ont été considérées :

- les supports internes (MEMØIRE, STOCK, BANDE)
- les supports externes (ECARTE, SCARTE, etc...)

Il est possible de placer en mémoire n'importe quel objet. Tout article ou toute file dont les éléments sont des objets simples fondamentaux (i.e. tout objet pouvant recevoir une valeur par une instruction d'attribution) peut être placé sur les autres supports internes.

Seuls les articles ou les files dont les éléments sont de nature MØT peuvent être placés sur un support externe.

Fondamentalement, le contenu d'un paquet de cartes, d'un ruban perforé, d'une bande magnétique, du stock, le message envoyé au clavier d'une machine à écrire, les lignes sorties sur une imprimante ont des structures de files.

De façon plus précise :

Support	Nombre d'entrées de la file	Nature de l'article
Cartes	1	1 mot de longueur 80 (ou 120)
Ruban perforé	1	(1)
Clavier	1	(1)
Papier	(2)	
Imprimé	(3)	
Bande magnétique	1	(4)
Stock	1	(4)

REMARQUES

- (1) un caractère ou une suite de caractères de longueur variable entre deux marques.
- (2) la file peut comprendre deux entrées pour le traitement :
- la 1ère entrée est relative aux lignes;
 - la 2ème entrée est relative aux colonnes.

La file peut ne comprendre qu'une entrée, dans ce cas l'index est relatif aux colonnes.

De toute façon les opérations élémentaires d'entrée-sortie ne transmettant qu'une ligne; la file à deux entrées est une commodité de programmation.

- (3) la file peut comporter deux entrées :
- la 1ère entrée est relative aux lignes;
 - la 2ème entrée est relative aux colonnes.

La file peut ne comporter qu'une entrée, dans ce cas l'index est relatif aux colonnes.

L'utilisation d'une file à deux entrées n'est, comme ci-dessus, qu'une facilité de programmation, e.g. traitement des bulletins pré-imprimés.

- (4) un enregistrement de nature quelconque entre deux marques.

La nature du support élimine certaines organisations (2.9.2) ; dans le tableau suivant les associations possibles sont signalées par le caractère *

Support \ Organisation	Séquentiel	Séquentiel indexé	Partitionné	Direct
Cartes	*			
Ruban perforé	*			
Clavier	*			
Papier	*			
Imprimé	*			
Bande magnétique	*			
Stock	*	*	*	*

Le statut de l'objet impliqué dans une opération d'entrée-sortie introduit certaines contraintes ; dans le tableau suivant les associations possibles sont signalées par le caractère

Opération Statut	Lecture	Ecriture	Mise à jour ou modification
DONNEE	*		
MIXTE	*	*	*
RESULTAT		*	
AUXILIAIRE	*	*	*

Un article ou une file en mémoire est représenté avec sa longueur maximale ; autrement dit, toutes les zones variables (condition, répétitions) sont générées.

Un tel article ou une telle file peut, dans certains cas, occuper une place importante sur les supports externes ou internes (autres que MEMØIRE) C'est pourquoi on a introduit deux possibilités, visant à réduire soit l'encombrement du fichier ou le temps de traitement. Ces deux possibilités sont :

1. L'article ou la file placé sur un support autre que MEMØIRE est une image de l'article ou de la file placé dans la mémoire. Lorsqu'un tel objet est introduit en mémoire, alors il ne subit aucune transformation de structure, hormis les conversions éventuelles décrites par des phrases de type 4. Il en résulte une économie du temps de traitement.
2. L'article ou la file placé sur un support autre que MEMØIRE est condensé en ce sens qu'il ne contient que les objets qui sont logiquement présents ainsi qu'il est spécifié par les valeurs des paramètres des zones variables (conditions et répétitions). Lorsque l'utilisateur désire faire un quelconque traitement sur cet objet il lui faut alors "dilater" celui-ci avant toute chose ; en d'autres termes, il faut que cet objet redevienne une image mémoire. Cette façon d'opérer augmente le temps de traitement, mais diminue la place occupée par les fichiers sur les supports autres que MEMØIRE. Cela limite évidemment le choix des méthodes d'accès.

De même, les processus de conversion, décrits par les phrases de type 4, introduisent quelques limitations quant à ce choix des méthodes d'accès

L'utilisateur doit préciser, dans la description de son fichier, l'option qu'il a choisie.

2.9.12.1 - MACRO INSTRUCTION DCB

L'utilisateur doit définir le Data Control Block (DCB) dans le programme où le fichier est traité ou encore de se référer à un DCB défini dans le programme appelant (Macro-instruction DCBD).

La macro-instruction DCB permet de définir le DCB associé à un fichier. Elle est de la forme :

```
MACROØDCB, ( ' < nom de DCB > " , ' < nom de fichier > " ; ; )
      { , ( < caractéristique > , spécification > ) } . . . ;
```

où

< nom de DCB > est un identificateur dont les huit premiers caractères sont pris en compte, cet identificateur est repris dans l'ordre DD.

< nom de fichier > est un identificateur qui désigne un article ou une file dont les caractéristiques liées aux opérations d'entrée-sortie sont définies par la partie macro-instruction de l'ordre

Les paramètres de la partie macro-instruction sont directement adaptés de ceux de la macro-instruction DCB de l'Operating System 360 (11).

2.9.12.2 - OUVERTURE ET FERMETURE DES FICHIERS

2.9.12.2.1 - MACRO-INSTRUCTION OPEN

La macro-instruction ØPEN permet d'ouvrir un fichier. Elle est de la forme :

```
[ < étiquette > : [ { · étiquette > : { . . . } ] ] MACROØØPEN, ( ' < nom de DCB > " ; ; )
      { , ( < caractéristique > , < spécification > { . . . ; ·
```

< nom de DCB > permet de se référer au DCB associé au fichier qu'on veut ouvrir.

La partie macro-instruction permet de préciser :

- _ le mode de traitement (lecture, écriture, ...)
- _ le choix des opérations qu'on désire que le système fasse à la fin du traitement

2.9.12.2.2 - MACRO-INSTRUCTION CLØSE

La macro-instruction CLØSE permet de fermer un fichier. Elle est de la forme :

```
[ < étiquette > : [ { < étiquette > : { ... } } ] ] MACRØCLØSE , ( ' < nom de DCB " ; ; )
{ , ( < caractéristique > , < spécification > ) } ... ;
```

< nom de DCB > permet de se référer au DCB associé au fichier qu'on veut fermer

La partie macro-instruction permet de préciser :

- le repositionnement du volume,
- le type de la fermeture (e.g. fermeture temporaire)

2.9.12.2.2.1 - Macro-instruction FEØV

Cette macro-instruction crée des conditions analogues à celles de la fin de volume. Elle ne peut être employée qu'avec les méthodes d'accès QSAM et BSAM. Elle est de la forme

```
[ < étiquette > [ { < étiquette > : { ... } } ] ] MACRØFEØV , ( ' < nom de DCB > " ; ; ) ;
```

2.9.12.3 - ACQUISITION ET CONTROLE DES BUFFERS

2.9.12.3.1 - MACRO-INSTRUCTION BUILD

La macro-instruction BUILD permet de construire un pool de buffers. Ce pool de buffers est une file. Elle est de la forme :

```
[ < étiquette > : [ { < étiquette > : { ... } } ] ] MACRØBUILD , ( < nom de file > ; ; ) ;
```

< nom de file > désigne une file de structure particulière (elle doit comprendre le Buffer Control Block BUFCB).

2.9.12.3.2 - MACRO-INSTRUCTION GETPØØL

La macro-instruction GETPØØL permet de faire construire un pool de buffers par le système. La forme particulière de cette macro-instruction n'est pas encore définie.

Il en est de même pour les macro-instructions FREEPØØL et GETBUF.

La partie concernant le contrôle des buffers est à l'étude.

2.9.12.4 - TRAITEMENT DES ERREURS

2.9.12.4.1 - MACRO-INSTRUCTION SYNADAF

Lorsque l'utilisateur utilise cette macro-instruction, le système analyse automatiquement les causes d'erreur et produit un message sous forme d'une chaîne de caractères désignée par l'argument résultat. Elle est de la forme :

< étiquette > : [{ étiquette ~ } ...] MACROSYNADAF, (< paramètre ; ; < expqua >) ;

On précisera plus tard la forme de cette macro-instruction (paramètre et forme du résultat).

2.9.12.5 - TRAITEMENT DES FICHIERS SEQUENTIELS

Les macros-instructions suivantes permettent de traiter les fichiers séquentiels.

2.9.12.5.1 - QUEUED ACCESS TECHNIQUE - QSAM

- GET - Obtention d'un article
- PUT - Ecriture d'un article
- PUTX - Ecriture en mode mise à jour
- RELSE - Abandon d'un buffer d'entrée
- TRUNC - Troncation d'un buffer de sortie

2.9.12.5.2 - BASIC ACCESS TECHNIQUE - BSAM

- BSP - Recul d'un bloc physique (BANDE et STOCK)
- CHECK - Test de la fin d'une opération READ/WRITE
- NOTE - Obtention de la position relative d'un bloc
- POINT - Pointage sur un bloc
- READ - Lecture d'un bloc
- WAIT - Attente de la fin d'une opération READ/WRITE
- WRITE - Ecriture d'un bloc

2.9.12.5.3 - MACROS-INSTRUCTIONS COMMUNES

CNTRL - Contrôle des périphériques "on-line", par exemple :

Lecteur de cartes : sélection de case

Imprimante : saut suivant canal ou espaces

Bande magnétique : recul ou avance de la bande (en BSAM seulement).

FEØV - Création d'une fin de volume

PRTØV - Test canal pour débordement sur imprimante "on-line",

2.9.12.6 - TRAITEMENT DES FICHIERS SEQUENTIELS INDEXES

Les macros-instructions suivantes permettent de traiter les fichiers séquentiels indexés.

2.9.12.6.1 - QUEUED ACCESS TECHNIQUE QISAM

ESELT - Arrêt de la recherche séquentielle des articles

GET - Obtention d'un article

PUT - Ecriture d'un article

PUTX - Ecriture d'un article mis à jour

RELSE - Abandon d'un buffer d'entrée

SETL - Départ de la recherche séquentielle des articles à partir d'une limite donnée.

2.9.12.6.2 - BASIC ACCESS TECHNIQUE-BISAM

READ - Lecture d'un article (un bloc est lu)

WAIT - Attente de la fin d'une opération READ/WRITE

WRITE - Ecriture d'un article, ou d'un bloc mis à jour

2.9.12.7 - TRAITEMENT DES FICHIERS EN ACCES DIRECT

Les macro - instructions suivantes permettent de traiter les fichiers en accès direct. La "Basic access technique" est seule utilisable, la méthode d'accès étant la BDAM.

- CHECK - Test de la fin d'une opération READ/ WRITE
- READ - Lecture d'un bloc
- RELEX - Abandon du contrôle exclusif d'un bloc
- WAIT - Attente de la fin d'une opération READ/ WRITE
- WRITE - Ecriture d'un bloc

2.9.12.8 - TRAITEMENT DES FICHIERS PARTITIONNES

Les macros-instructions suivantes permettent traiter les fichiers partitionnés. La "Basic access technique" est seule utilisable, la méthode d'accès étant la BPAM.

- BLDL - Obtention du répertoire du fichier (directory) en mémoire.
- CHECK - Test de la fin d'une opération READ/ WRITE
- FIND - Pointage sur un membre
- NOTE - Obtention de la position d'un bloc
- POINT - Pointage sur un bloc
- READ - Lecture d'un bloc
- STOP - Modification du répertoire
- WAITE - Attente de la fin d'une opération READ/ WRITE

2.9.12.9 - COMMUNICATIONS AVEC L'OPERATEUR

- WTL - Ecriture d'un message sur la console système (MVT)
- WTØ - Ecriture d'un message à l'opérateur
- WTØR - Ecriture d'un message à l'opérateur avec demande de réponse.

2.10 - MACRO - INSTRUCTIONS DU COMPILATEUR

Les macro-instructions ont été introduites essentiellement pour permettre à l'utilisateur d'exprimer les ordres d'entrée-sortie. Mais leur forme très générale permet de définir de nouvelles opérations qui n'étaient pas prévues dans la première version du langage.

Ces nouvelles macro-instructions ont été appelées : macro-instructions du compilateur. En fonction de leur usage on les classe en 4 catégories :

1. Macro-instructions de compilation
2. Macro-instructions d'aide à la mise au point (Debug)
3. Macro-instructions définies par l'utilisateur
4. Services du système d'exploitation.

Nous allons examiner successivement chacune de ces catégories.

2.10.1 - Macro-instructions de compilation

Les macro-instructions de compilation, qui appartiennent à l'une des deux classes suivantes :

- macro-déclarations,
- macro-directives,

permettent à l'utilisateur d'utiliser certaines commodités d'exploitation et de programmation offertes par le compilateur. Un petit nombre de ces macro-instructions a été défini ; on peut aisément l'augmenter en fonction des besoins.

2.10.1.1 - INSERTION

Il est possible de demander au compilateur de remplacer l'appel d'une procédure par l'insertion de celle-ci dans le programme en cours de compilation.

Le compilateur effectue la substitution des paramètres formels et adjoint les descriptions des auxiliaires (de la procédure insérée) à celles du programme en cours de compilation.

Pour de plus amples détails, voir L. NOLIN dans "Formalisation des notions de machine et de programme" (21).

La division d'un traitement en un programme du type "Traitement immédiat" et une série de procédures donne une structure logique à l'ensemble. Ceci contribue à accélérer la mise au point.

Toutefois, cette division conduit à une légère augmentation du temps de traitement. L'utilisateur peut demander l'insertion de quelques procédures afin d'accélérer le traitement.

Le gain n'est d'ailleurs pas aussi important qu'il peut apparaître à première vue ; en effet sitôt qu'un article ou une file occupe plus de 4096 octets (dans la version IBM 360 du langage) les programmes doivent avoir recours à l'indexation pour adresser un élément.

La forme de cette macro-instruction (en fait, macro-directive) est :

$$\left[\begin{array}{l} \langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \left\{ \dots \right\} \right\} \dots \right] \end{array} \right] \langle \text{nom de procédure} \rangle , \left(\left[\begin{array}{l} \langle \text{argd} \rangle \left[\left\{ , \langle \text{argd} \rangle \left\{ \dots \right\} \right\} \right] \right] ; \left[\begin{array}{l} \langle \text{argm} \rangle \left[\left\{ , \langle \text{argm} \rangle \left\{ \dots \right\} \right\} \right] \right] ; \left[\begin{array}{l} \langle \text{argr} \rangle \left[\left\{ , \langle \text{argr} \rangle \left\{ \dots \right\} \right\} \right] \right] \end{array} \right] \right) ;$$

la substitution est requise par la macro-instruction INSERT en lieu et place de cet appel :

$$\left[\begin{array}{l} \langle \text{étiquette} \rangle : \left[\left\{ \langle \text{étiquette} \rangle : \left\{ \dots \right\} \right\} \right] \end{array} \right] \text{MACROINSERT}, \left(\left[\begin{array}{l} \langle \text{argd} \rangle \left[\left\{ , \langle \text{argd} \rangle \left\{ \dots \right\} \right\} \right] \right] \right] ; \left[\begin{array}{l} \langle \text{argm} \rangle \left[\left\{ , \langle \text{argm} \rangle \left\{ \dots \right\} \right\} \right] \right] ; \left[\begin{array}{l} \langle \text{argr} \rangle \left[\left\{ , \langle \text{argr} \rangle \left\{ \dots \right\} \right\} \right] \right] \end{array} \right] \right) , \left(\text{NAME} , \langle \text{nom de} \right. \\ \left. \text{procédure} \rangle \right) ;$$

Il est nécessaire que le nom de la procédure soit connu au moment de la compilation, e.g. :

DONNEE & PROCEDURE & PROC1 ;

AUXILIAIRE & ARTICLE & APPEL & & PARTAP & & ;

AUXILIAIRE & PROCEDURE & PROC2 ;

AUXILIAIRE & NATUREL & IND & 2 & ;

PARTAP & & & 5 & PROCEDURE & 8 & CALL ;

~~PROC2~~ & SALAIRE ;

CALL □ DE □ APPEL & TAUX1, TAUX2, TAUX3, TAUX4, TAUX5 ;

① ~~PROC1~~, (____ ; ____ ; ____) ;

② ~~PROC2~~, (____ ; ____ ; ____) ;

③ CALL (1) DE APPEL, (____ ; ____ ; ____) ;

④ CALL (IND) DE APPEL, (____ ; ____ ; ____) ;

Les expressions suivantes sont autorisées pour l'insertion :

② ~~MACRO~~ INSERT, (____ ; ____ ; ____) , (NAME, ~~PROC2~~) ;

③ ~~MACRO~~ INSERT, (____ ; ____ ; ____) , (NAME, CALL (1) DE APPEL) ;

par contre les expressions suivantes sont interdites :

① ~~MACRO~~ INSERT, (____ ; ____ ; ____) , (NAME, ~~PROC1~~) ;

④ ~~MACRO~~ INSERT, (____ ; ____ ; ____) , (NAME, CALL (IND) DE APPEL) ;

car le nom des deux procédures est inconnu au moment de la compilation

2.10.1.2 - MISE EN PARALLELE DES PARTITIONS

La description en parallèle des partitions (phrases de type 4) est fastidieuse et peut ne pas être très lisible. L'utilisateur peut demander, grâce à la macro-description PARAL, au compilateur de réaliser cette mise en parallèle des descriptions des partitions en phrases de type 3 ou 4, ou à partir des descriptions cataloguées.

La forme de cette macro-instruction est :

```
MACROØ PARAL, ( < nom de partition 1 > , < nom de partition 2 > [ { , < nom de partition i >
                                     } ... ] ; ; ) ;
```

< nom de partition 1 > est le nom de la partition mise en parallèle avec < nom de partition 2 > et éventuellement avec plusieurs autres partitions < nom de partition i >.

Les conditions auxquelles doivent satisfaire les partitions et leurs éléments sont les mêmes que celles énoncées à propos de la description des phrases de type 4 (2.4.4).

Les macro-instructions PARAL doivent être placées entre les phrases de type 4 et celles de type 5 d'un programme.

2.10.1.3 - CATALOGAGE DES PARTITIONS

La première version du langage prévoyait que les partitions décrites dans un traitement immédiat étaient automatiquement cataloguées. Si donc on voulait supprimer la partition du catalogue, alors il fallait recourir à l'instruction EFFACER.

Il a semblé plus expédient d'introduire la notion de "catal à la demande" grâce à la macro-directive CATAL. Cette macro-instruction permet, si on fait figurer une séquence de noms en argument, de cataloguer simultanément plusieurs partitions.

De même, la macro-directive DELCATAL (DELETE CATAL) permet de supprimer des descriptions de partition du catalogue.

Ces macro-instructions doivent être placées entre les phrases de type 4 et celles de type 5 d'un programme. Les opérations sont effectuées dans l'ordre de leur apparition. e.g.

C DERNIERE DES PHRASES DE TYPE 3 ØU 4

C

```
MACROØ DELCATAL, (PART1 ; ; ) ;
```

```
MACROØ CATAL, (PART1 ; ; ) ;
```

C

PREMIERE DES PHRASES DE TYPE 5 (ØU 6)

C

dans cet exemple PART1 est supprimé du catalogue et est remplacé par sa nouvelle description.

La forme et l'organisation du catalogue des partitions n'étant pas figées actuellement, il est impossible de décrire précisément ces macros-instructions.

2.10.1.4 - ENTREES-SORTIES STANDARD

Les instructions d'entrée-sortie exposées au paragraphe précédent sont très complètes, mais leur complexité peut rebuter l'utilisateur de prime abord. Aussi, a-t-on prévu de définir certaines méthodes d'accès standard ; elles offrent de moins grandes possibilités et elles conduisent à un traitement moins efficace. Dans ces méthodes, les opérations sur les fichiers s'expriment à l'aide d'ordres simplifiés.

Ces macro -instructions sont définies pour les quatre types d'organisation de fichier, i.e

- séquentiel
- séquentiel indexé
- direct
- partitionné

et les deux techniques d'accès :

- Queued Access Technique
- Basic Access Technique

L'utilisateur n'a dans ce cas, à sa disposition, qu'un nombre réduit de macro -instructions, e.g

GETST

PUTST

READST

WRITEST

OPENST

CLOSEST

qui le ramène à des conditions peut être plus habituelles.

Cependant ces macro -instructions gardent la forme classique, et peuvent susciter l'emploi des ordres complets.

2.10.1.5 - POINTS DE REPRISE

Deux macro -instructions seront définies à ce propos, CHKPT et RESTART. La macro-instruction CHKPT (check point) permet de créer sur disque un fichier permettant la reprise éventuelle du traitement à ce point.

La macro-instruction RESTART n'est utilisable que dans un traitement immédiat, elle permet de reprendre l'exécution depuis un point de reprise créé par CHKPT

2.10.2 - Macro -instructions d'aide à la mise au point

Le compilateur peut, à partir des indications fournies par des macros-instructions, créer des ordres d'impression de certains objets afin de faciliter la mise au point.

Actuellement deux types de macro-instructions ont été prévus à cet effet, TRACE et SNAP

2.10.2.1 - TRACE

L'utilisation de la macro-instruction TRACE dans une procédure, permet à l'utilisateur de provoquer, au cours de l'exécution de cette procédure, l'impression de certains objets à différentes étapes du traitement. C'est-à-dire :

- l'impression de certaines étiquettes rencontrées au cours du traitement
- l'impression de certains objets.

Le format est :

```
MACRO TRACE, ( [ < variable de condition > ] ; ; ) , ( < étiquette > , < variable >
[ { , < variable > { ... } } ] [ { , ( < étiquette > , < variable > [ { < variable > ... } ] { ... } } ] ;
```

où < variable de condition > définie par :

$$\left. \begin{array}{l} < expqua > \\ < identificateur > \end{array} \right\}$$

désigne un nombre booléen conditionnant la sortie de la trace, à chaque fois que l'une des étiquettes mentionnées est rencontrée lors du traitement.

< étiquette > est une étiquette de la procédure, la rencontre de cette étiquette dans le traitement, si la condition est satisfaite, provoque l'impression de < étiquette > et des valeurs des objets désignés par la liste de variables associée à < étiquette >.

La variable est imprimé en regard de sa valeur.

< variable > définie par :

$$\left\{ \begin{array}{l} \text{< expqua >} \\ \text{< identificateur >} \end{array} \right\}$$

désigne un objet dont la valeur est à imprimer.

Il est possible de placer plusieurs macros-instructions TRACE dans un même programme. Elles sont placées entre les phrases de type 4 et celles de type 5 du programme

2.10.2.2 - SNAP

L'utilisation de la macro-instruction SNAP permet à l'utilisateur de faire imprimer, à l'exécution, une partie des informations placées en mémoire.

Les profils hexadécimaux des zones sont imprimés. Il y a quatre types de macro-instruction SNAP selon les informations à imprimer :

1. SNAPARG - impression de tout ou partie des arguments d'une procédure
2. SNAPAUX - impression de tout ou partie des auxiliaires d'un programme
3. SNAPIØ - impression des informations associées au traitement des fichiers
4. SNAPCØDE - impression de tout ou partie du code généré du programme

Les deux derniers types, SNAPIØ et SNAPCØDE, ont été introduits surtout pour faciliter la mise au point du compilateur.

Le format général est :

$$\left[\text{< étiquette >} : \left[\left\{ \text{< étiquette >} : \{ \dots \} \right\} \right] \text{MACRØSNAP} \left\{ \begin{array}{l} \text{ARG} \\ \text{AUX} \\ \text{IØ} \\ \text{CØDE} \end{array} \right\}, (\left\{ \begin{array}{l} \text{< variable 1 >} \\ \text{' < chaîne de caractères >} \end{array} \right\} \right)$$

$$\left[, < \text{variable de condition} > \right] ; ;) \left[\left\{ , (\text{LIST}, < \text{variable} > \left[\left\{ , < \text{variable} > \right\} \dots \right] \right) \right\} \dots \right]$$

avec

< variable 1 > désigne un mot qui sera placé en tête des informations imprimées.

< chaîne de caractères > " même signification que < variable 1 >.

< variable de condition > désigne un nombre booléen conditionnant la sortie du SNAP.

Selon de type de SNAP, les paramètres figurant après LIST diffèrent :

1. SNAPARG

< variable > désigne un argument (i.e un objet de statut DØNNEE, MIXTE ou RESULTAT) dont la valeur doit être imprimée, la place de cette variable ne peut être que MEMØIRE.

Dans le cas où LIST est omis, tous les arguments de support MEMØIRE seront imprimés.

2. SNAPAUX

< variable > désigne un objet de statut AUXILIAIRE et de support MEMØIRE dont la valeur doit être imprimée.

Dans le cas où LIST est omis, tous les objets de statut AUXILIAIRE et de support MEMØIRE seront imprimés.

3. SNAPIØ

< variable > désigne un article ou une file faisant l'objet d'une opération de changement de support.

Les caractéristiques des opérations d'échange (DCB, etc...) sont imprimés.

Dans le cas où LIST est omis, toutes les caractéristiques des articles et des files, faisant l'objet d'opérations de changement de support, sont imprimées.

4. SNAPCØDE

Le format est :

$$\left[< \text{étiquette} > : \left[\left\{ < \text{étiquette} > \right\} \dots \right] \text{MACRØSNAPCØDE}, \left(\begin{array}{l} < \text{variable 1} > \\ ' < \text{chaîne de caractères} > \end{array} \right) \right. \\ \left. , < \text{variable de condition} > \right] ; ;) \left[\left\{ , (\text{LIST}, < \text{étiquette 1} > , < \text{étiquette 2} >) \right\} \dots \right] ;$$

< étiquette 1 > et < étiquette 2 > sont des étiquettes du programme.

Le code machine des instructions depuis < étiquette 1 > jusqu'à < étiquette 1 > est imprimé.

Dans le cas où LIST est omis, le code machine de toutes les instructions du programme est imprimé.

2.10.3 - Macro -instructions définies par l'utilisateur

Le langage permettant d'écrire les prototypes des macro-instructions n'est pas défini actuellement.

La définition de ce langage est l'une des prochaines extensions ; on s'inspirera, pour cela, des formats des prototypes de macro-instructions utilisés par le compilateur

2.10.4 - Services du système d'exploitation

Le système d'exploitation met une grande variété de macro -instructions à la disposition des utilisateurs pour la gestion des tâches et de la mémoire notamment. La quasi totalité de ces macro -instructions ne peut être utilisée qu'au niveau des programmes de type "traitement immédiat" (2.11).

Le seul service utilisable par les procédures, pour le moment, est la donnée du "temps" par le système d'exploitation à l'aide de la macro-instruction TIME.

La forme est :

$$\left[\text{< étiquette >} : \left[\left\{ \text{< étiquette >} : \{ \dots \} \right\} \right] \text{MACRO TIME,} \left(\left\{ \begin{array}{l} \text{'DEC'} \\ \text{'BIN'} \\ \text{'TU'} \end{array} \right\} ; ; \text{< variable >} \right) ;$$

où < variable > est un identificateur ou une expression qualifiée dont la nature dépend de l'option (DEC, BIN ou TU) choisie :

1. DEC - l'objet désigné par < variable > doit être un nombre décimal de format 6.2 ou 8.0; l'heure du jour est obtenue sous la forme :

HHMMSS [] DC

avec

HH heure (24 de base)

MM minutes

SS secondes

DC dixièmes et centièmes de seconde

2. BIN - l'objet désigné par < variable > doit être un nombre naturel ou un nombre entier de longueur 9 (représenté sur 4 octets). L'heure du jour est obtenue sous forme d'un nombre naturel représentant l'heure exprimée en centièmes de seconde.

3. TU - l'objet désigné par < variable > doit être un nombre naturel ou entier représenté sur 4 octets.

l'heure du jour est exprimée en multiples de 24 micro -secondes.

'DEC" est l'option par défaut.

Le statut de l'objet désigné par < variable > doit permettre la présence de l'objet parmi les arguments RESULTAT d'un appel de procédure (2.8.5).

2.11 - TRAITEMENT IMMEDIAT ET PROCEDURE

La matière dont les procédures sont utilisées montre qu'elles ne sont rien d'autre que des instructions, plus complexes que les autres, et n'appartenant pas, comme elles, à un répertoire immuable.

L'utilisation des procédures pour réaliser un traitement spécifique nécessite, entre autres choses, la description des données qui leur serait fournies, la réservation de la place nécessaire aux résultats qu'elles produiront. Tout ceci entre dans le cadre des **traitements immédiats**.

La notion de traitement immédiat diffère peu de celle de **PRØGRAMME** qui a été introduite dans le langage ATF de base (21).

Un traitement immédiat s'exprime par les mêmes phrases que les procédures. Son nom est un identificateur dont les premiers caractères sont :

TRAITEMENT IMMEDIAT ...

De ce fait, aucun nom de procédure ne peut commencer par ce mot.

Par rapport à la première version du langage, la présence de phrases de type 1 est autorisée pour la commodité de compilation et d'exécution (recherche des procédures appelées).

Les objets décrits par des phrases de type 2, sont tous de statut **AUXILIAIRE**, et les descriptions ont pour premier objet de réserver la place qui leur est nécessaire.

Contrairement à ce qui se passe pour les procédures, les objets décrits dans un traitement immédiat, qui sont tous de statut **AUXILIAIRE**, peuvent être placés sur des supports externes. Ceci revient implicitement à décrire ces objets comme étant de statut **DØNNEE**, **MIXTE** ou **RESULTAT** en fonction du support considéré et des caractéristiques de la méthode d'accès.

Les descriptions obéissent à toutes les règles édictées pour les descriptions d'objets auxiliaires dans les procédures, sauf en ce qui concerne la place et les longueurs. Les longueurs, fixées pour la représentation des objets, doivent être connues au moment de la compilation (constantes ou longueurs standard par défaut) pour pouvoir réserver la place nécessaire.

Les définitions (phrases de type 5) ont le même effet que dans les procédures.

Dans la première version du langage, les instructions (phrases de type 6) comportaient exclusivement :

- des instructions d'attribution ;
- des ordres de changement de support ;

- des instructions, réservées aux traitements immédiats, dont l'effet est de libérer la place occupée par certains objets périmés :

EFFACER □ < nature > □ < identificateur >

où < identificateur > désigne l'objet à supprimer dont la nature est indiquée par < nature >.

Les traitements immédiats se déroulaient de façon strictement séquentielle : les instructions de saut étaient interdites et les étiquettes sans utilité.

Dans la version actuelle du langage, les instructions se composent uniquement des instructions ou macro-instructions suivantes :

- appels de procédures ;
- macro-instructions pour la description des fichiers ;
- macro-instructions pour la gestion des ressources (services du système d'exploitation).

et de l'instruction FIN, obligatoire dans tout programme.

L'instruction EFFACER est conservée actuellement, elle ne s'applique qu'à des objets isolés ou des objets groupés placés en mémoire.

Les macro -instructions relatives aux services moniteur qu'il est possible de faire figurer dans un traitement immédiat sont fonctions du système particulier sur lequel le langage est adapté, elles ne sont pas décrites ici.

2.12 - CONCLUSION

Tel qu'il est défini actuellement, le langage possède une grande parenté avec un langage machine. Il en a la simplicité mais présente des avantages certains ; énumérons quelques points :

- les descriptions sont des macro-réservations ; elles permettent, non seulement de réserver la place nécessaire à la représentation de l'objet ou de son adresse, mais d'associer un type (e.g. NATUREL) à l'objet décrit ;
- les définitions sont des macro-équivalences ; elles permettent d'attribuer une valeur constante à certains objets du traitement, la variable devenant le synonyme de la constante ;
- les procédures correspondent à des traitements bien définis ; on peut imaginer que leurs arguments correspondent aux parties adresses des instructions ;
- les instructions d'attribution sont de forme très simple, en général à 3 adresses (sauf une qui est la division décimale avec facteur de reste) tout comme en langage machine (e.g. IBM 360 souvent 3 adresses : 2 adresses explicites, une adresse implicite) ;
- le traitement immédiat est destiné à prendre la place du langage de commande (cartes contrôle).

Le grand avantage que procure le langage est la structure logique qu'il confère aux programmes, ce que ne fait évidemment pas un langage machine classique et même certains langages symboliques récents dits de haut niveau.

Un assez grand nombre de modifications a été apporté à la première version du langage (22).

Ces modifications visent à rendre le langage plus directement utilisable et plus efficace tout en gardant la cohérence de sa définition.

L'apport le plus marquant est certainement celui des macro-instructions. Elles sont une façon commode d'exprimer des opérations complexes ; ces opérations complexes auraient nécessité, en l'absence des macro-instructions, plusieurs instructions pour leur expression ; en général :

- une suite d'instructions d'affectation ;
- un ou plusieurs appels de procédure ;
- une suite d'instructions de saut conditionnel ;
- une suite d'instructions d'affectation.

Les macro-instructions permettent de résumer toutes ces opérations en une seule instruction ; elles sont un heureux complément des procédures.

Les macro-instructions sont destinées à évoluer en fonction des besoins, notamment de ceux que pourront faire apparaître les opérations d'entrée-sortie d'une part et d'autre part les services que le compilateur et le système d'exploitation peuvent rendre à l'utilisateur.

Remarquons que certaines macro-instructions, les macro-directives, les macro-descriptions et les macro-équivalences permettent d'augmenter de façon considérable les possibilités du langage puisque les opérations auxquelles ils correspondent ne faisaient pas partie du langage.

Les macro-instructions permettent d'exprimer les opérations de changement de support notamment ; celles-ci ont été complètement modifiées par rapport à la première version du langage. Elles permettent de plus grandes possibilités de traitement des fichiers associées à une meilleure efficacité tout en gardant une interface simple avec le système d'exploitation et une mise au point relativement aisée de ces ordres.

La forme d'expression de ces opérations de changement de support permet d'admettre facilement de nouveaux ordres. Ces nouveaux ordres peuvent être définis pour avoir une expression plus simple et une mise en oeuvre moins délicate, en général ceci est au détriment de l'efficacité.

Parmi les extensions projetées, mentionnons :

- l'introduction d'un caractère séparateur dans les identificateurs e.g. : \ ;
- la substitution d'un caractère spécial au mot réservé DE dans les expressions de qualification, qui ressembleraient alors à celles du PL 1. Ainsi :

BULLETIN DE PAIE & DE EMPLOYE (I) & DE USINE FABRICATION & DE PARIS
deviendrait :

BULLETIN_Δ DE_Δ PAIE . EMPLOYE (I) . USINE_Δ FABRICATION_Δ PARIS

- une syntaxe différente - visant à supprimer le caractère séparateur & - qui permettrait une écriture plus aisée des programmes

Une autre extension consiste à introduire les expressions arithmétiques, e.g. :

$A := (((B + C) / D) + F) * G) ** E - 0.25 ;$

Une telle extension entraîne la définition de nouvelles règles arithmétiques sur les natures et les largeurs des objets qui figurent dans l'expression. Le langage perdrait, avec cette adjonction, quelque peu en simplicité, et surtout la parenté qu'il possède avec les langages machine.

L'extension la plus importante sera certainement celle de la notion de macro-instruction, elle débouchera sur la définition d'un langage permettant à l'utilisateur de définir ses prototypes de macro-instruction.

3 — COMPILATEUR

3.1 - INTRODUCTION

Le premier point important dans la définition du compilateur, et aussi pour certains aspects du langage, est le choix de la machine.

La présente étude aboutissant d'une part à la définition du langage et d'autre part à la description du compilateur ainsi qu'au début de sa programmation, il importe que la suite des travaux puisse se faire dans de bonnes conditions. En outre, une fois ce compilateur terminé, il faut intéresser le plus grand nombre d'utilisateurs au langage. Ce langage étant orienté vers les applications de gestion, ceci définit, à priori, une gamme de matériels. D'autre part pour que les travaux se poursuivent aisément, il faut que le type d'ordinateur retenu soit facilement accessible.

Tout ceci laisse le choix entre deux modèles :

IBM 360

CII 10070

Cependant, le fait que l'IBM 360 est plus répandu que le CII 10070 est déjà un élément de la décision. En outre, au départ de cette étude, le moniteur de multiprogrammation du 10070 était en cours d'élaboration. Un autre élément en faveur de l'IBM 360 est la présence du compilateur du langage ATF de base sur cette machine.

La réalisation a donc été entreprise sur IBM 360.

Le système d'exploitation retenu est l'Operating System (ØS) avec l'option MFT (Multiprogramming with a Fixed number of Tasks) ou MVT (Multiprogramming with a Variable number of Tasks).

Au départ il était prévu de programmer le compilateur en langage ATF de base, le compilateur ATF de base ayant été programmé en ATF de base et généré par "bootsprapping"

Cette technique donne de bons résultats ; elle permet de réaliser relativement rapidement un compilateur, en général au détriment de l'efficacité.

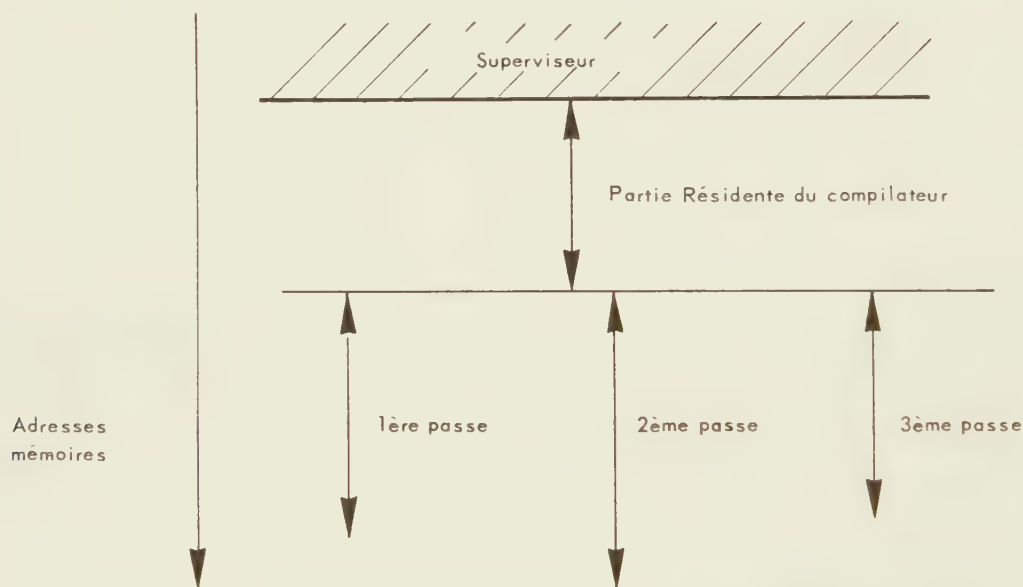
Par définition le langage étant orienté vers le traitement des problèmes de gestion, cela impose des contraintes en fonction des utilisateurs possibles. Il est nécessaire d'avoir un compilateur occupant une place aussi réduite que possible en mémoire, et donnant des temps de compilation semblables à ceux du CØBØL ou du FØRTRAN par exemple. Les temps

d'exécution doivent être de l'ordre de grandeur de ceux des langages existants pour le traitement de problèmes similaires (et si possibles inférieurs ...). Le compilateur doit parfaitement s'intégrer au système et produire des programmes (modules objets) standards.

On a choisi de réaliser un compilateur en plusieurs "passes". La "passe" est prise ici dans le sens où les programmes du compilateur qui la constituent modifient le texte proposé à l'entrée.

Du fait des contraintes de place, volontairement imposées, on a retenu une organisation avec une partie résidente, et 3 phases de traitement (chacune réalisant une "passe") s'exécutant consécutivement (sans retour à la phase précédente).

Cette division permet d'adopter une structure d'"overlay" où phases s'exécutent en recouvrement : seule la partie résidente reste, évidemment, en mémoire lors de la compilation, les trois passes étant exécutées par 3 phases appelées successivement et chargées pour exécution au même emplacement mémoire, e.g.



De plus cette architecture permet, si la capacité mémoire est suffisante, d'introduire en mémoire simultanément deux ou trois phases.

Les files intermédiaires créées par les "passes" du compilateur peuvent très bien être stockées temporairement sur disque ou sur bande dans les cas de capacités mémoire très faibles.

La division du compilateur en phases bien séparées donne une structure logique à l'ensemble et de ce fait crée de bonnes conditions de mise au point; les "passes" ont des relations entre elles uniquement par les files intermédiaires qu'elles génèrent et par les informations stockées dans la partie résidente.

De plus cette structure permet de prendre facilement en compte les modifications du langage, ce qui est une aide précieuse pour un nouveau langage.

Le langage de programmation retenu pour l'écriture du compilateur est le langage Assembleur IBM 360 (4).

On l'a retenu en raison du codage très fin (utilisation du code machine et considération des temps d'exécution) qu'il permet, de la possibilité qu'il offre pour traiter des files de format très particulier, et de la souplesse d'emploi qu'il procure avec ses macro-instructions.

En outre, l'usage du langage d'assemblage permet de disposer de tous les services du système d'exploitation, ce que les langages de programmation existants ne permettaient pas.

Un autre avantage du codage en assembleur est de pouvoir chiffrer, avec précision, du point de vue :

- Temps de compilation ;
- Place occupée en compilation ;
- Temps d'exécution ;
- Place occupée en exécution

le coût de certaines facilités du langage.

Comme il est dit ci-dessus, le compilateur est conçu de façon à être compatible avec le système d'exploitation de l'IBM 360. En conséquence, il doit pouvoir être "appelé" comme tout autre compilateur (FORTRAN, COBOL, PL/1, etc...) et il doit produire des modules objets (object modules) compatibles avec l'éditeur de liens (Link Edit).

En fonction de ces contraintes l'organisation générale est la suivante :

1. Phase résidente ou racine

Cette partie du compilateur est résidente en mémoire durant toute la compilation. Elle a pour but de rassembler toutes les informations nécessaires (options spécifiées pour le compilateur, manipulation de catalogues de partitions, etc...) et d'organiser l'enchaînement des "passes" du compilateur.

2. Phase d'analyse syntaxique

Cette phase réalise l'analyse syntaxique du texte source. Elle produit :

- une file objet qui est un "recodage" du texte source reconnu ;
- une file de messages d'erreur éventuellement ;

- une série d'informations à l'usage de la seconde "passe" du compilateur et de la racine, notamment pour la recherche des partitions cataloguées et des procédures à insérer.

3. Phase de pré-génération et de vérification

Cette phase travaille à partir des données contenues dans la racine, et recherchées puis introduites par la racine en mémoire éventuellement, et d'autre part de la file objet générée par la 1ère passe (analyse syntaxique).

Elle produit :

- une file objet qui est en quelque sorte un pseudo-code machine ;
- une file de messages d'erreur éventuellement ;
- une série d'informations à l'usage de la troisième "passe" et de la racine.

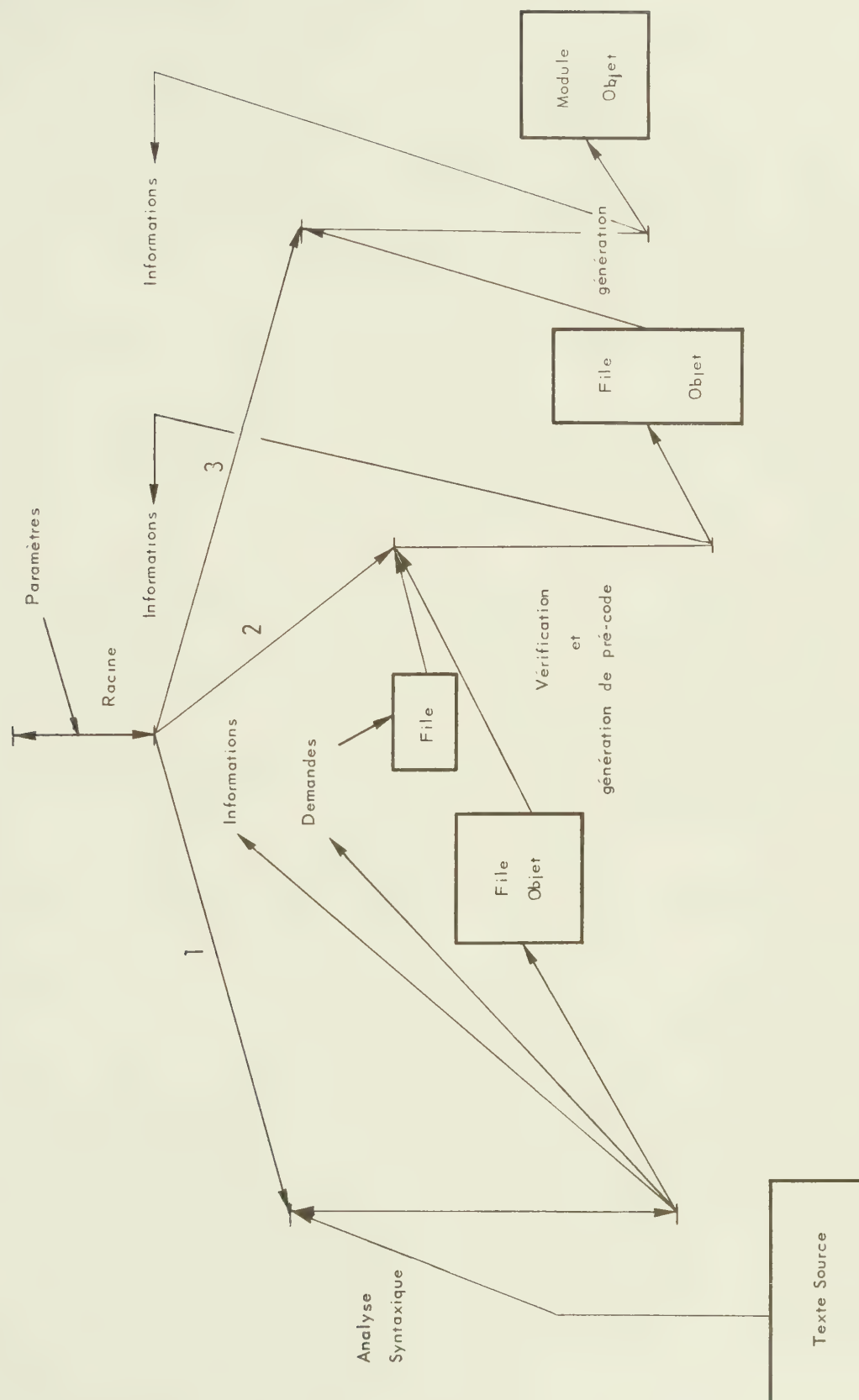
4. Phase de génération

Cette phase travaille à partir des données contenues dans la racine et dans la file objet générée par la deuxième "passe" (vérification et génération de pré-code).

Elle produit :

- un module objet ;
- une file de messages d'erreur éventuellement ;
- une série d'informations à l'usage de la racine.

En résumé, l'organisation générale du compilateur peut se schématiser de la façon ci-après.



On a introduit un certain nombre d'options pour l'ordre de commande pour l'ordre de commande qui spécifie la compilation (ordre EXEC) :

- 1 - NLS Pas de liste du texte source (No Listing Source)
- 2 - LC Listing des partitions cataloguées
- 3 - LM Listing du pseudo-code généré par certains macros-instructions
- 4 - LØ Listing du code machine généré pour le programme
- 5 - XREF Listing des tables des références croisées
- 6 - MAP Géographie des données
- 7 - SEQ Vérification de séquence

3.2 - PHASE D'ANALYSE SYNTAXIQUE

Le choix d'une méthode d'analyse syntaxique s'impose ; parmi les diverses solutions, on a retenu la méthode d'analyse descendante ou "top-down"

Cette méthode d'analyse est utilisée depuis longtemps par L. NOLIN. Dans le cadre de la CII, B. KAHAN et H. DUMAS-PRIMBAULT ont entrepris une étude pour la mise en pratique de cette technique. Ces travaux ont abouti à la définition d'une forme "externe" de grammaire et ils ont permis d'écrire un programme qui génère l'automate correspondant à la grammaire donnée

Les fondements théoriques ont été présentés par D.E. KNUTH (17). Le programme permettant la génération des automates est rédigé en FORTRAN. Dans la forme initiale des grammaires proposées à ce programme, les noms de règles ou d'actions étaient remplacés par des numéros, ce qui présente une gêne pour l'utilisateur, il a donc été nécessaire de modifier ce programme pour admettre des noms de règles en place des numéros. La méthode adoptée pour la formulation des règles ainsi que le programme d'analyse (interprétation de l'automate) sont exposés successivement.

3.2.1 - Formalisme adopté pour la description syntaxique

Il sera présenté à l'aide d'exemples.

1. Soient les règles suivantes exprimées à l'aide des notations utilisées pour l'ALGOL (BACKUS NAUR Form ou BNF) :

$$\langle \text{prog} \rangle ::= \langle \text{instruc} \rangle$$

$$\langle \text{instruc} \rangle ::= V = \langle E \rangle ;$$

$$\langle E \rangle ::= \langle T \rangle \mid \langle T \rangle + \langle E \rangle ;$$

$$\langle T \rangle ::= \langle P \rangle \mid \langle P \rangle * \langle T \rangle$$

$$\langle P \rangle ::= (\langle E \rangle) \mid V \mid N$$

Cette notation utilise les métasymboles suivants :

- les chevrons < et > qui délimitent les noms de symboles syntaxiques non terminaux ;
- le symbole ::= avec la signification de "défini par" ;
- le signe $\mid$ avec la signification de "ou".

Les autres symboles rencontrés dans une règle syntaxique sont des symboles terminaux. Le deuxième membre d'une règle est formé par une chaîne de symboles terminaux et non terminaux, ou par plusieurs de ces chaînes séparées par le métasymbole $|$. En conséquence chacune de ces chaînes est une détermination du symbole non terminal situé à gauche du signe $::=$ et ce symbole est défini par cette règle.

L'introduction de nouveaux métasymboles permet de réécrire les règles précédentes sous la forme :

$$\begin{aligned} \langle \text{prog} \rangle & ::= \langle \text{instruc} \rangle \\ \langle \text{instruc} \rangle & ::= \langle T \rangle \Sigma \} + \langle T \rangle \{ ; \\ \langle T \rangle & ::= \langle P \rangle \Sigma \} * \langle P \rangle \{ \\ \langle P \rangle & ::= (\langle E \rangle) \mid V \mid N \end{aligned}$$

Les métasymboles introduits sont :

- les métaparenthèses $\}$ et $\{$ qui permettent de réunir provisoirement une chaîne de symboles terminaux et non-terminaux ou plusieurs de ces chaînes séparées par $|$ en un symbole composé unique. Un symbole composé se comporte comme un symbole non terminal unique vis à vis des symboles syntaxiques qui n'en font pas partie.

Lorsqu'un symbole composé comporte plusieurs chaînes séparées par $|$, ces chaînes sont des déterminations du symbole composé. Une détermination d'un symbole composé peut comprendre à son tour des symboles composés.

- le métasymbole Σ qui indique que le symbole syntaxique simple ou composé qui le suit immédiatement, peut se reproduire un nombre indéfini de fois ou ne pas exister du tout.

L'introduction du métasymbole Σ permet de remplacer certaines des définitions récurrentes par des définitions itératives.

Ces nouvelles notations introduisent plus de souplesse dans la formulation des règles syntaxiques, notamment pour les analyseurs "top-down" dans lesquels les récursions à gauche sont interdites.

L'utilisation de ces nouvelles notations dans l'écriture des règles précédentes accélère le déroulement de l'analyse :

$$\begin{aligned} \langle \text{prog} \rangle & ::= \langle \text{instruc} \rangle \\ \langle \text{instruc} \rangle & ::= \langle V \rangle = \langle E \rangle ; \end{aligned}$$

$$\begin{aligned}
\langle E \rangle & ::= \langle P \rangle \Sigma \{ + \langle T \rangle \mid *' \langle P \rangle \} \\
\langle T \rangle & ::= \langle P \rangle \Sigma \{ *' \langle P \rangle \} \\
\langle P \rangle & ::= (\langle E \rangle) \mid V \mid N
\end{aligned}$$

à partir de cette notation, une dernière transformation doit être faite pour arriver à la formulation utilisée dans la méthode :

$$\begin{aligned}
\langle \text{PROG} \rangle & ::= \langle \text{INSTR} \rangle \downarrow \\
\langle \text{INSTR} \rangle & ::= V = \langle E \rangle ; \downarrow \\
\langle E \rangle & ::= \langle P \rangle \Sigma \{ + \langle T \rangle \mid *' \langle P \rangle \} \downarrow \\
\langle T \rangle & ::= \langle P \rangle \Sigma \{ *' \langle P \rangle \} \downarrow \\
\langle P \rangle & ::= (\langle E \rangle) \mid V \mid N \downarrow
\end{aligned}$$

Le nouveau métasymbôle $\downarrow$ indique la fin de règle.

Le métalangage comporte en outre les métasymboles suivants :

Δ - représentant la chaîne vide (ou la chaîne de longueur nulle ou mot vide), e.g. :

Soit la règle :

$$\langle \text{REGLE} \rangle ::= AB \mid A \downarrow$$

en factorisant elle devient :

$$\langle \text{REGLE} \rangle ::= A \{ B \mid \Delta \} \downarrow$$

- ⊕ - qui, suivi d'un identificateur, peut être placé à la fin d'une détermination du symbole syntaxique défini par la formule ou d'un symbole composé. Il correspond à l'ordre d'exécution d'une procédure type "fin de règle" permettant, en particulier, des actions sémantiques.

L'identificateur est le nom de la procédure. Ce type de symbole est appelé "Insertion vraie".

- ⊙ - qui, suivi d'un identificateur, peut être placé après un symbole soit composé, soit unique. Dans le cas où le premier élément de ce symbole n'est pas reconnu, la procédure désignée par l'identificateur est exécutée.

Ce type de symbole est appelé "Insertion fausse".

- qui suivi d'un identificateur et d'un ' ', correspond à l'ordre inconditionnel d'exécution d'une procédure.

Ce type de symbole est appelé "séquence d'actions directe"

Ces notations métalinguistiques permettent de décrire la structure d'un langage en définissant le processus de génération, à partir d'un symbole syntaxique particulier < AXIØME >, des chaines terminales appartenant au langage considéré.

La partie analyse syntaxique d'un compilateur pose le problème inverse qui est la recherche du mode de génération, à partir du symbole < AXIØME >, d'un texte source donné.

Dans ce but, il est nécessaire de modifier la signification des symboles syntaxiques et des métasymboles de façon à pouvoir interpréter la syntaxe du langage comme un programme d'analyse.

Ce programme est constitué par une suite de procédures récurrentes représentées par les règles syntaxiques. Pour chaque règle syntaxique le nom figurant à gauche du symbole $::=$ est le nom de la procédure et la chaine de symboles à droite de ce symbole jusqu'à, et y compris, le symbole $\downarrow$ est le corps de la procédure.

Les symboles syntaxiques et les métasymboles $\downarrow$, $\oplus$, $\odot$ et $\cdot$ sont interprétés comme des instructions de la procédure, les métasymboles $|$, $\{$, $\}$ et Σ interviennent dans la détermination de l'ordre d'exécution de ces instructions.

3.2.2 - Signification des instructions

Soit $S_1, S_2, \dots, S_i, \dots$ la chaine de symboles source à analyser, S_i désignant le symbole en cours d'examen et $U_1, U_2, \dots, U_k$ la chaine de symboles constituant les instructions du corps de la procédure en cours d'exécution (les métasymboles $|$, $\{$, $\}$ et Σ sont ignorés dans cette numérotation de symboles, $\odot$ joue un rôle particulier). Si n est le nom de la procédure en cours d'exécution, une instruction est identifiée par le doublet (n, k) .

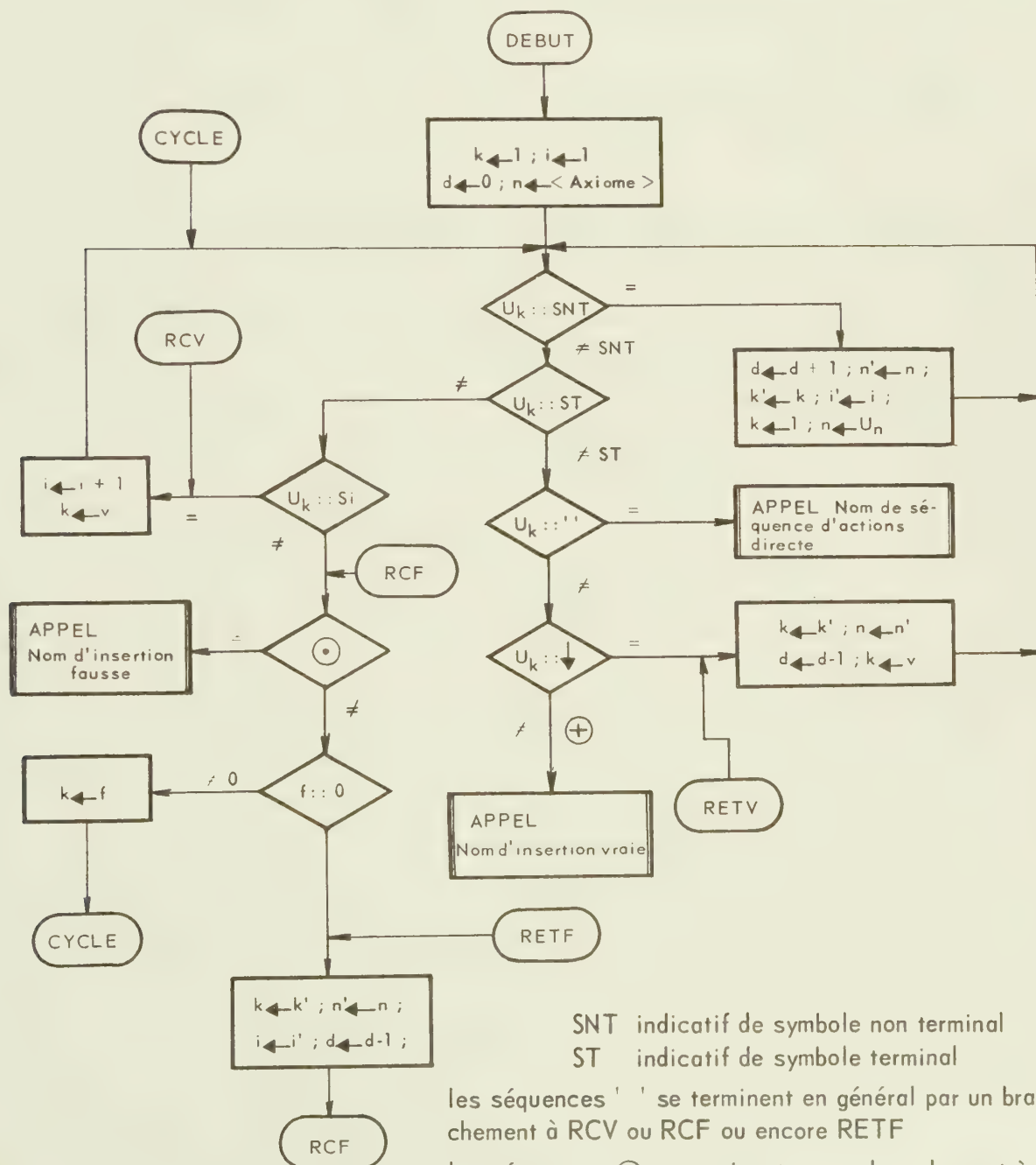
Pour définir l'ordre d'exécution des instructions du programme d'analyse, on associe à tout symbole U_k , lorsque c'est un symbole syntaxique, le symbole U_v qui est le successeur de U_k dans la même détermination du symbole défini par la règle, et le symbole U_f qui est l'"alternant" de U_k appartenant à une autre détermination.

Pour un symbole syntaxique U_k donné, U_v existe toujours par contre U_f peut ne pas exister ; la valeur de f est alors nulle par convention.

Pour une procédure donnée v et f sont des fonctions de k . La définition des successeurs et des alternants est donnée au paragraphe 3.2.3.

L'exécution d'un tel programme nécessite la mémorisation de certaines informations. Celle-ci est faite à l'aide d'une pile d'organisation LIFO (Last In First Out), i.e. le dernier élément introduit dans la pile est le premier à en sortir, soient $P_1, P_2, \dots, P_e$ les informations contenues dans la pile P_e désignant toujours le dernier élément introduit. Cette pile contient essentiellement trois valeurs, n', k' et i' , qui sont respectivement : le nom de la procédure précédente, le numéro de l'instruction dans cette procédure et le caractère du texte source (ou plutôt la valeur d'index associée à ce texte source) en cours d'examen lorsque ces informations ont été mises en pile.

L'organigramme de l'analyseur est :



Cet organigramme est équivalent au programme suivant, rédigé dans un langage inspiré de l'ALGØL :

C

C DEBUT DE L'ANALYSE

C

DEBUT : $k \leftarrow 1$; $i \leftarrow 1$; $d \leftarrow 0$; $n \leftarrow \text{AXIØME}$;

C

C Exécution d'une instruction de l'automate

C

CYCLE : IF ($U_k = \text{symbole non terminal}$) THEN GØ TØ BRANCH ELSE GØ TØ SUITE1 ;

C BRANCH correspond à l'appel d'une procédure (règle).

C Entrée dans la pile:

BRANCH : $d \leftarrow d + 1$;

$n' \leftarrow n$; $k' \leftarrow k$; $i' \leftarrow i$; $k \leftarrow 1$;

$n \leftarrow U_k$; GØ TØ CYCLE ;

C Suite de procédure d'exécution

SUITE1 : IF ($U_k \neq \text{Symbole terminal}$) THEN GØ TØ SUITE2 ;

C U_k représente un symbole terminal

C S_i est le caractère pointé dans le texte source.

IF ($U_k = S_i$) THEN GØ TØ RECOÏNU ELSE GØ TØ RACCØRD-FAUX ;

C Le caractère S_i est reconnu

RECOÏNU : $i \leftarrow i + 1$; $k \leftarrow v'$;

GØ TØ CYCLE ;

C Le caractère S_i n'a pas été reconnu

RACCØRD-FAUX : IF (U_k possède une insertion fausse $\odot$) THEN CALL Nom d'insertion fausse ;

IF ($f = 0$) THEN GØ TØ RETØUR-FAUX ;

C U_k possède un alternant

$k \leftarrow f$;

GØ TØ CYCLE ;

C U_k ne possède pas d'alternant

C Sortie d'un élément de la pile

RETØUR-FAUX : $k \leftarrow k'$; $n \leftarrow n'$; $i \leftarrow i'$; $d \leftarrow d-1$; GØ TØ RACCØRD-FAUX ;

C Continuation de la procédure d'exécution

SUITE 2 : IF ($U_k =$ Séquence d'action directe) THEN CALL Nom de séquence directe ;

C Fin de règle ou insertion vraie ($\oplus$)

IF ($U_k =$ fin de règle) THEN GØ TØ RETOUR-VRAI ;

C Insertion vraie ($\oplus$)

CALL Nom d'insertion vraie ;

C Sortie d'un élément de la pile

RETØUR-VRAI : $k \leftarrow k'$; $n \leftarrow n'$; $d \leftarrow d-1$;

$k \leftarrow N$;

GØ TØ CYCLE ;

Les séquences d'actions directes (i.e. ' Nom de séquence ') se terminent en général par l'une des instructions :

GØ TØ RECØNNU ;

GØ TØ RACCØRD-FAUX ;

GØ TØ RETØUR-FAUX ;

Les insertions vraies (i.e. $\oplus$ < Nom d'insertion >) se terminent par l'une des instructions :

$G\emptyset \ T\emptyset \ RET\emptyset UR-VRAI ;$

$G\emptyset \ T\emptyset \ RET\emptyset UR-FAUX ;$

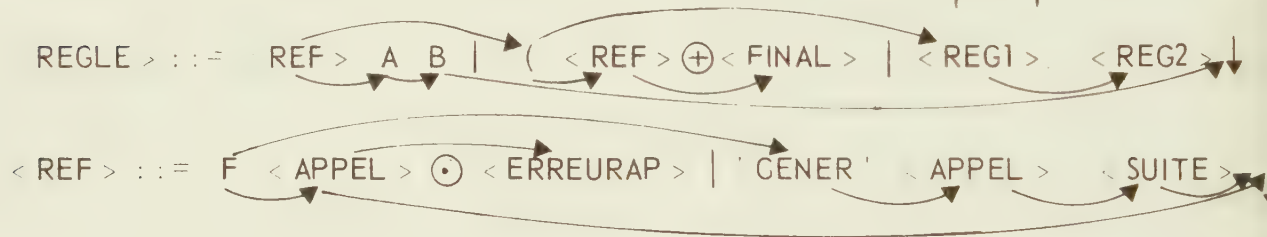
$ST\emptyset P ;$

Dans le cas des insertions fausses (i.e. $\odot$ < Nom d'insertion >), il n'est pas possible de déterminer, dans le cas général, le retour de la séquence.

3.2.3 - Définition des symboles successeur (U_v) et alternant (U_f) d'un symbole U_k donné

La manière la plus simple de procéder, est d'utiliser des exemples.

Soit, en premier lieu, le cas des règles sans les métasymboles Σ , $\{$ et $\}$. e.g.



L'origine de chaque flèche définit un symbole U_k et l'extrémité des flèches :

- du bas, le symbole successeur associé U_v ;
- du haut, le symbole alternant associé U_f ;

Pour un symbole syntaxique U_k , différent de $\oplus$ ou de $\odot$, le successeur U_v existe toujours. C'est le symbole suivant dans la même détermination du symbole défini par la règle, sauf pour le dernier symbole syntaxique d'une détermination lorsque ce symbole est suivi par le métasymbole $|$. Dans ce cas le successeur est le métasymbole $\downarrow$.

L'alternant U_f peut exister pour les symboles syntaxiques qui suivent les métasymboles $::=$ ou $|$, i.e. pour les premiers symboles de chaque détermination. Pour ces symboles, l'alternant est le premier symbole de la détermination suivante, si elle existe.

Dans le cas d'une insertion fausse ($\odot$), le symbole précédant cette insertion à celle-ci pour alternant. L'écriture :

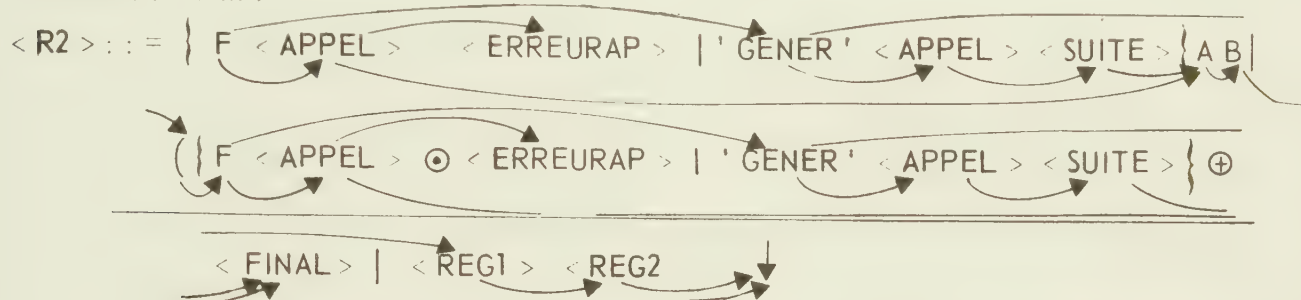
$\langle REG \rangle ::= \leq A \rangle \langle B \rangle | \langle C \rangle \odot \langle ERC \rangle \langle D \rangle | \langle E \rangle \langle F \rangle \downarrow$

est incorrecte puisque $\langle C \rangle$ a deux alternants, $\langle E \rangle$ et $\langle ERC \rangle$, ce qui est interdit.

Par convention f est nul pour tous les symboles qui n'ont pas d'alternant.

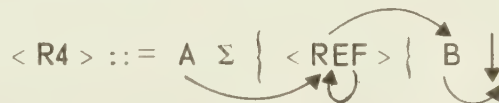
Dans les règles précédentes, c'est le cas des symboles A, B, < REF >, < REG1 >, < REG2 >, < APPEL >, < SUITE > qui sont soit des symboles non en tête d'une détermination ou en tête de la dernière détermination.

Pour passer au cas de définitions comportant des symboles composés, il suffit de remplacer dans l'exemple précédent, le symbole < REF >, dans la règle < REGLE >, par sa définition. Il vient :



Le symbole successeur de < REF > dans la règle < REGLE > (1ère et 2ème détermination) devient dans la règle < R2 > le successeur du dernier symbole de chaque détermination du symbole composé qui a remplacé le symbole < REF >.

Soit la détermination suivante qui comprend le métasymbole Σ :

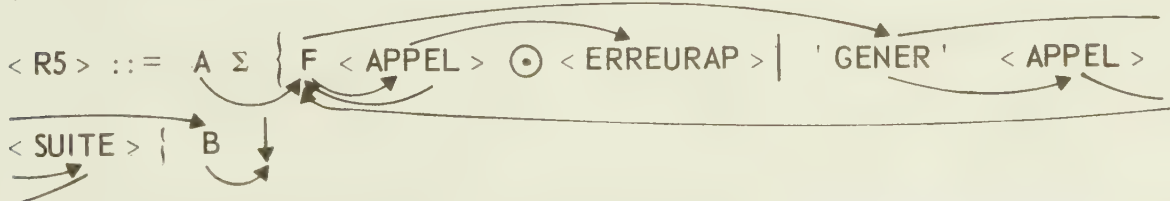


Le successeur du symbole syntaxique qui précède Σ est déterminé en ignorant la présence de Σ .

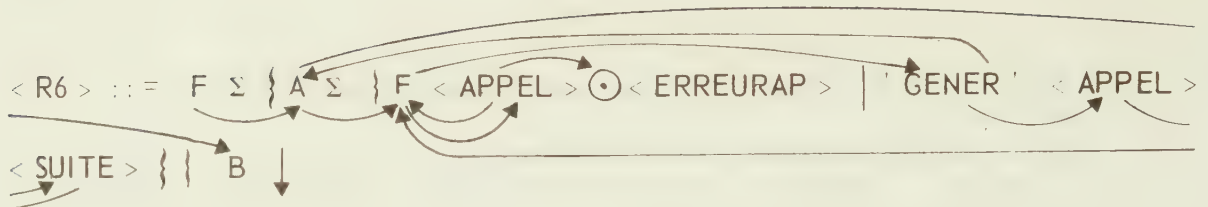
Le successeur du symbole syntaxique qui est immédiatement précédé d'un Σ est ce symbole lui-même.

L'alternant du symbole syntaxique qui est immédiatement précédé d'un Σ est le symbole qui serait le successeur du symbole considéré s'il n'était pas précédé d'un Σ .

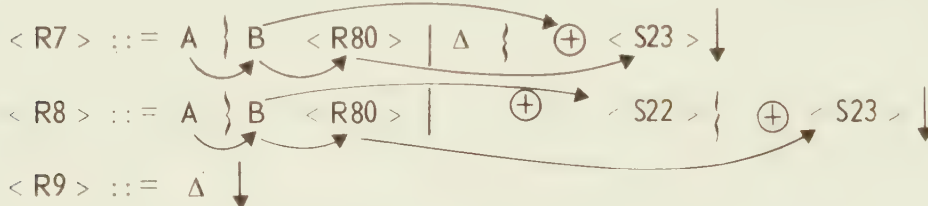
Soit maintenant la règle < R5 > obtenue à partir de < R4 > en remplaçant < REF > par sa définition :



Les flèches définissant le successeur et l'alternant des différents symboles s'obtiennent sans difficulté en tenant compte des indications qui précèdent. Il en est de même pour la règle suivante :



Les exemples suivant ont trait à l'emploi du symbole Δ représentant le mot vide :



La procédure représentée par cette règle se réduit à l'exécution de l'instruction représentée par $\downarrow$, i.e. : $G\emptyset T\emptyset RET\emptyset UR-VRAI$;

A partir des indications précédentes, il est facile de voir que la définition de la syntaxe interprétée comme programme d'analyse permet d'effectuer l'analyse des chaînes de caractères constituées de symboles terminaux. Cette méthode utilise de principe d'analyse "top-down", en conséquence, une mise en forme préalable de la grammaire est nécessaire. Les conditions de validité du programme d'analyse ont été précisées par KNUTH (17).

3.2.4 - Introducteur de syntaxe

Les séquences du type "insertion vraie" permettent de faire certaines vérifications sémantiques correspondant à des fins de règles.

Les séquences du type "insertion fausse" sont utilisées pour le traitement des erreurs détectées dans le texte source.

Les séquences directes (' ') permettent de faire des vérifications sémantiques, de générer un texte codé ou de vérifier directement une partie du texte source (e.g. mots réservés).

Il est nécessaire d'opérer une transformation sur la syntaxe du langage pour obtenir un code facilement interprétable par l'ordinateur. C'est l'objet du programme introducteur de syntaxe "CHARADE" rédigé en FØRTRAN.

Ce programme travaille à partir de la définition de la grammaire et génère l'automate correspondant sous forme de définitions de constantes (DC) en langage d'assemblage IBM 360.

La grammaire du langage est transcrite dans une forme plus aisément traitable par le programme e.g. :

Soit la règle :

$$\langle \text{REGLE} \rangle ::= \{ \langle \text{REF} \rangle \text{ AB } | \text{ F } \} \Sigma \{ \text{ D } ' \text{ GEND } ' \} \square \odot \langle \text{ERBLANC} \rangle \{ | \text{ E } \} \oplus \langle \text{VERIF} \rangle \downarrow$$

elle devient :

$$\text{REGLE} = @ (@ \text{RREF} \text{ AB } @ | \text{ F } @ (@ \text{S} @ (\text{D } @ \text{D} \text{ GEND } @) \mathcal{C} @ \text{J} \text{ERBLANC } @) @ | \text{ E } @) @ \text{I} \text{VERIF} @ \text{F}$$

Le caractère blanc $\square$ a été remplacé par le caractère $\mathcal{C}$ du fait que les blancs ne sont pas significatifs dans un tel format sauf pour délimiter les noms des non-terminaux dans le cas où ceux-ci comportent moins de 8 caractères.

Les règles de substitution sont :

$::=$	$\longrightarrow$	$=$
$\langle \rangle$	$\longrightarrow$	@ R sauf pour la définition d'une règle, e.g. : $\langle \text{REGLE} \rangle ::= \longrightarrow \text{REGLE} \square =$
$\{$	$\longrightarrow$	·@ (
$\}$	$\longrightarrow$	@)
Σ	$\longrightarrow$	@ S
$ $	$\longrightarrow$	@
	$\longrightarrow$	@ D e.g. 'DIR' $\longrightarrow$ @ DDIR
$\oplus$	$\longrightarrow$	@ I
$\odot$	$\longrightarrow$	@ J
$\downarrow$	$\longrightarrow$	@ F
$\square$	$\longrightarrow$	$\mathcal{C}$

Ce programme opère une vérification de la syntaxe donnée, cette vérification ne porte pas sur les conditions citées par KNUTH (17), mais seulement sur la possibilité de construire un automate à partir de la syntaxe. L'une des prochaines extensions est la réalisation d'un programme permettant la vérification des conditions de KNUTH.

A chaque règle correspond une liste, l'automate se présente donc comme une suite de listes. Chaque mot de liste correspond à une instruction, à chaque symbole syntaxique (sauf $\odot$) est associé un mot de liste comprenant 5 parties : b, p, m, v et f.

La signification de ces parties est la suivante :

- pour chaque symbole terminal le contenu de la partie P, dite préfixe, est nul ; la partie m contient la représentation machine du symbole.
- pour un symbole non terminal (< >) p = 1 ; m contient, en place du nom de règle, le numéro d'ordre de la procédure.
- pour une séquence d'action directe (' ') p = 3 ; m contient, en place du nom de cette séquence, le numéro d'ordre de celle-ci.

Les parties v et f contiennent respectivement les raccords aux mots de liste associés au successeur et à l'alternant du symbole considéré.

Si l'alternant n'existe pas f = 0

Dans le cas où le symbole syntaxique est immédiatement suivi d'une insertion fautive ($\odot$), b = 1 et f contient, en place du nom de cette séquence, le numéro d'ordre de celle-ci.

Aux métasymboles $\oplus$ et $\downarrow$ correspond un mot de liste comportant 3 parties : p, m et s avec p = 2, m contient le nom de la procédure (numéro d'ordre), indication qui n'est utilisée (actuellement) que pour la mise au point du compilateur.

Ces mots marquent la fin d'une détermination du symbole défini par la règle.

Pour le métasymbole :

- $\oplus$, s contient, en place du nom de la séquence, le numéro d'ordre de celle-ci.
- $\downarrow$, s = 0.

Les contraintes de place font qu'il est nécessaire d'utiliser les adresses relatives pour repérer les mots de liste relatifs au successeur et à l'alternant d'un symbole donné (contenus de v et f du mot de liste associé à ce symbole).

La représentation, en mots de liste, de la syntaxe suivante :

$$\begin{aligned} < \text{PROG} > ::= < \text{INSTR} > \oplus < \text{GEN} > \downarrow \\ < \text{INSTR} > ::= \text{V} = < \text{E} >; \odot < \text{ERRPTVIR} > \downarrow \end{aligned}$$

$$\begin{aligned}
\langle E \rangle &::= \langle P \rangle \Sigma \{ + \langle T \rangle \mid * \langle P \rangle \} \downarrow \\
\langle T \rangle &::= \langle P \rangle \Sigma \{ * \langle P \rangle \} \downarrow \\
\langle P \rangle &::= (' \text{GENPAR}' \langle E \rangle) \mid V \mid N \downarrow
\end{aligned}$$

est :

< PROG >	1 :	<table><tr><td>b</td><td>p</td><td>m</td><td>v</td><td>f</td></tr><tr><td>0</td><td>1</td><td>2</td><td>1</td><td>0</td></tr></table>	b	p	m	v	f	0	1	2	1	0	< INSTR >
b	p	m	v	f									
0	1	2	1	0									
		<table><tr><td></td><td></td><td></td><td>s</td><td></td></tr><tr><td>0</td><td>2</td><td>1</td><td></td><td>1</td></tr></table>				s		0	2	1		1	$\oplus$ < GEN >
			s										
0	2	1		1									
< INSTR >	2 :	<table><tr><td>0</td><td>0</td><td>V</td><td>1</td><td>0</td></tr></table>	0	0	V	1	0	V					
0	0	V	1	0									
		<table><tr><td>0</td><td>0</td><td>=</td><td>1</td><td>0</td></tr></table>	0	0	=	1	0	=					
0	0	=	1	0									
		<table><tr><td>0</td><td>1</td><td>3</td><td>1</td><td>0</td></tr></table>	0	1	3	1	0	< E >					
0	1	3	1	0									
		<table><tr><td>1</td><td>0</td><td></td><td>1</td><td>1</td></tr></table>	1	0		1	1	; $\odot$ < ERRPTVIR >					
1	0		1	1									
		<table><tr><td>0</td><td>2</td><td>2</td><td></td><td>0</td></tr></table>	0	2	2		0	$\downarrow$					
0	2	2		0									
< E >	3 :	<table><tr><td>0</td><td>1</td><td>5</td><td>1</td><td>0</td></tr></table>	0	1	5	1	0	< P >					
0	1	5	1	0									
		<table><tr><td>0</td><td>0</td><td>+</td><td>1</td><td>2</td></tr></table>	0	0	+	1	2	+					
0	0	+	1	2									

		b	p	m	v	f	
		0	1	4	- 1	0	< T >
		0	0	*	1	2	*
		0	1	5	- 3	0	< P >
		0	2	3		0	↓
< T >	4 :	0	1	5	1	0	< P >
		0	0	*	1	2	*
		0	1	5	- 1	0	P >
		0	2	4		0	↓
< P >	5 :	0	0	(	1	4	(
		0	3	1	1	1	GENPAR '
		0	1	3	1	0	< E >

b	p	m	v	f	
0	0	)	3	0	)
0	0	V	2	1	V
0	0	N	1	0	N
0	2	5	0		

Le programme CHARADE "introduceur de syntaxe" permet de faire cette génération d'automate. Il est nécessaire de choisir un format machine pour les mots de liste.

Les mots de liste sont tous de 4 octets, les différents champs se répartissent ainsi :

b	1 bit
p	2 bits
m	13 bits
signe de v	1 bit
v	7 bits
signe de f	1 bit
f	7 bits

13 bits ont été retenus pour l'expression de m en raison de la grandeur possible des informations (le nombre maximal pouvant être représenté dans m est 8191) ; lorsque p = 1, m contient l'adresse de la règle invoquée dans l'automate, de même lorsque p = 2, m contient l'adresse de la règle dont ce symbole marque la fin (ou la fin d'une détermination).

Lorsque p = 0, m contient la valeur du caractère (8 bits cadrés à droite dans cette zone) à reconnaître dans le texte source. Lorsque p = 3, m contient le numéro d'ordre de la séquence d'actions directe, ce qui permet de retrouver l'adresse de celle-ci en indexant dans un répertoire par la valeur de m.

v et f (dans le cas où $b = 0$) représentent sous forme d'un nombre entier exprimé en valeur absolue sur 7 bits, le 1er bit étant l'indication du signe (1 pour négatif, 0 pour positif), la valeur du déplacement dans l'automate pour atteindre le successeur et l'alternant.

Lorsque $p = 2$ et avec $v \neq 0$, v représente le numéro d'ordre de l'insertion vraie, ce qui permet d'en retrouver l'adresse par indexation dans un répertoire.

Lorsque $b = 1$, f représente le numéro d'ordre de l'insertion fautive, l'adresse de celle-ci étant obtenue par indexation dans un répertoire.

L'interprétation de l'automate exige la présence de trois répertoires correspondants aux :

- insertions vraies ($\oplus$)
- insertions fausses ($\ominus$)
- séquences d'actions directes (' ').

L'introducteur de syntaxe génère, outre certaines listes commodes pour la mise au point de la syntaxe, les ordres en langage d'assemblage IBM 360 correspondant à l'automate :

DC X' $\Delta \Delta \Delta \Delta \Delta \Delta \Delta \Delta$ '

et aux 3 répertoires

DC A ()

L'Annexe 2 contient un exemple de grammaire du langage ainsi que différentes sorties du programme CHARADE.

3.2.5 - Analyseur

L'analyseur est un programme qui effectue l'analyse d'un texte source, en utilisant la représentation définie ci-dessus des règles syntaxiques du langage. Il produit un texte codé représentant les phrases du texte source et éventuellement des messages d'erreur.

La génération de ce texte codé est faite à l'aide des séquences d'actions directes et des insertions vraies. Les messages d'erreur sont essentiellement produits par les insertions fausses et éventuellement par les séquences d'actions directes et les insertions vraies.

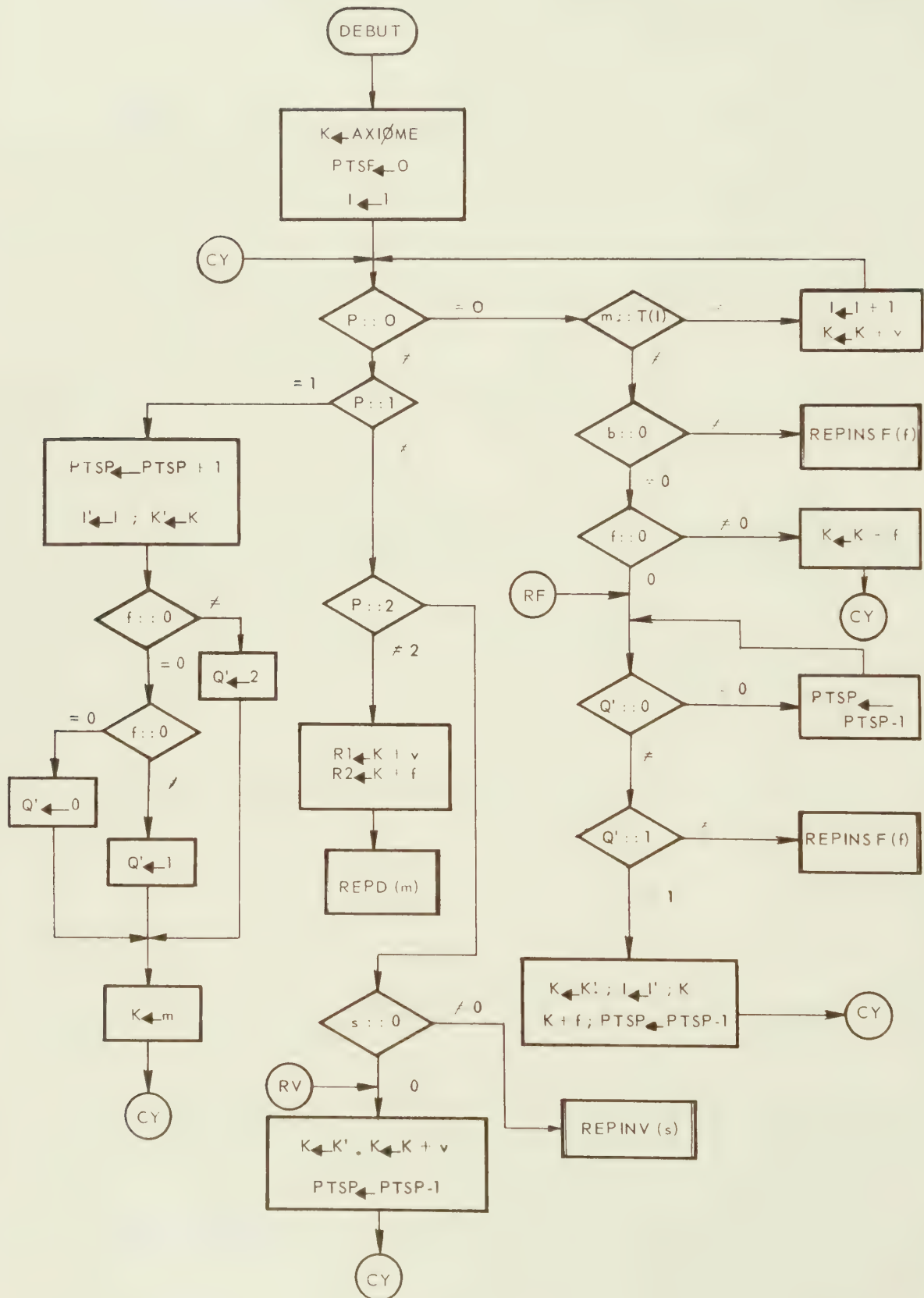
Toutes ces séquences sont rédigées en langage d'assemblage.

A partir de la nouvelle forme des mots de liste, il est nécessaire de définir le programme permettant l'interprétation de ceux-ci.

L'analyseur utilise les éléments suivants :

- une file à une entrée dans laquelle est rangé le texte source. En fait l'accès de cette file passe par une petite procédure dont le but est d'éliminer les cartes commentaire ('C' en colonne 1) et de limiter l'analyse à la colonne 72 (72 comprise). Cette procédure, si l'option SEQ est présente (3.1), vérifie la séquence des numéros d'identification portés dans les colonnes 73 à 80. Soit T cette file.
- un mot I dont le contenu indique le numéro d'ordre du caractère du texte source en cours d'examen. I est l'index de la file T.
- un mot K dont le contenu est l'adresse de l'instruction (mot de liste) de l'analyseur en cours d'exploitation. Les lettres b, p, m, v, f et s représentent toujours les parties du mot de liste pointé par K.
- une pile SP dont l'adresse du sommet est contenue dans le mot PTSP. Chaque mot de pile comporte, essentiellement, 3 parties désignées par l', K' et Q'. Ces lettres se rapportent toujours aux parties du mot de pile pointé par PTSP.
- Trois files, à une entrée chacune, REPINSV, REPINSF et REPD contenant respectivement les adresses des séquences d'insertions vraies, d'insertions fausses et d'actions directes.

L'organigramme de l'analyseur est :



R1 et R2 représentent les adresses de retour pour les séquences d'actions directes.

La grammaire du langage a été rédigée de façon simple, elle ne permet pas de faire des vérifications sémantiques importantes lors de l'analyse des instructions bien que les déclarations soient groupées en tête du programme source.

La présence des partitions cataloguées et des macros-instructions, qui se trouvent parmi les phrases de type 6, compliquerait énormément la phase d'analyse syntaxique dans ce cas. Il serait nécessaire d'opérer toutes les vérifications des déclarations avant de continuer l'analyse (à partir des phrases de type 5).

En conséquence, la grammaire n'est pas modifiée, à partir des phrases de type 5, pour pouvoir opérer des vérifications sémantiques.

Durant l'analyse des déclarations, les noms des objets du traitement sont entrés dans un dictionnaire constitué à l'aide d'une technique de Hash-Code (3.2.6).

La technique que l'on a retenue pour le calcul du Hash-code est utilisée, à la CII, dans la rédaction du compilateur CØBØL IRIS 50.

3.2.6 - Technique de "Hashing"

Le rangement des identificateurs déclarés dans les phrases de type 1, 2, 3 et 4 en une table organisée séquentiellement, implique en moyenne un parcours de la moitié de cette table pour la recherche d'un élément.

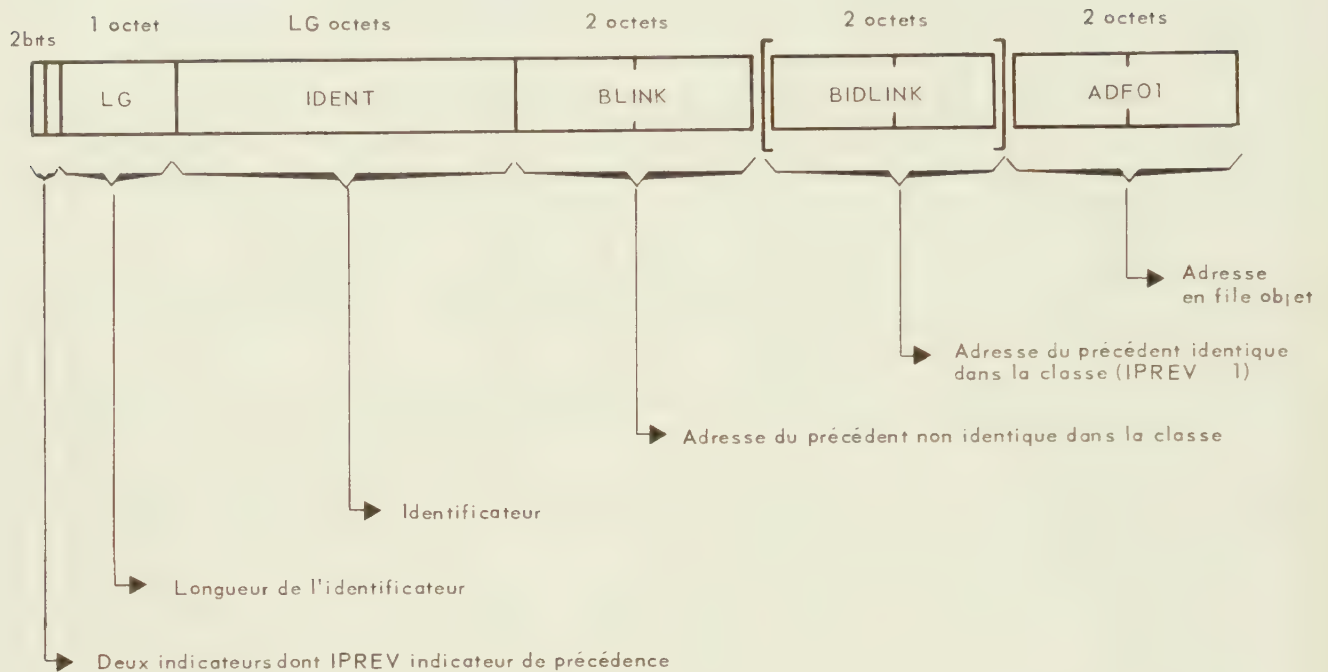
La technique de "Hashing" permet de réduire le champ d'exploitation pour la recherche en la restreignant à celle d'un sous-ensemble contenant l'élément.

Il s'agit de définir à l'intérieur de l'ensemble des éléments à ranger un certain nombre de sous-ensembles ou classes d'éléments. Cette partition est réalisée par l'intermédiaire d'un algorithme approprié (3.2.6.2) ; quatre paramètres sont alors associés à chaque élément de l'ensemble :

1. un nombre caractéristique de la classe à laquelle il appartient ;
2. l'adresse, dans la file générée par le programme d'analyse syntaxique, à laquelle cet élément est apparu ;
3. l'adresse de l'élément non identique qui le précède dans cette classe ;
4. éventuellement l'adresse de l'élément identique qui le précède.

La recherche en table d'un élément se fait en deux temps : recherche de la classe de l'élément, puis de l'élément à l'intérieur de sa classe.

Un élément du dictionnaire se présente de la façon suivante :



Soit HDICØ ce dictionnaire.

3.2.6.1 - CONSTITUTION DE HDICØ

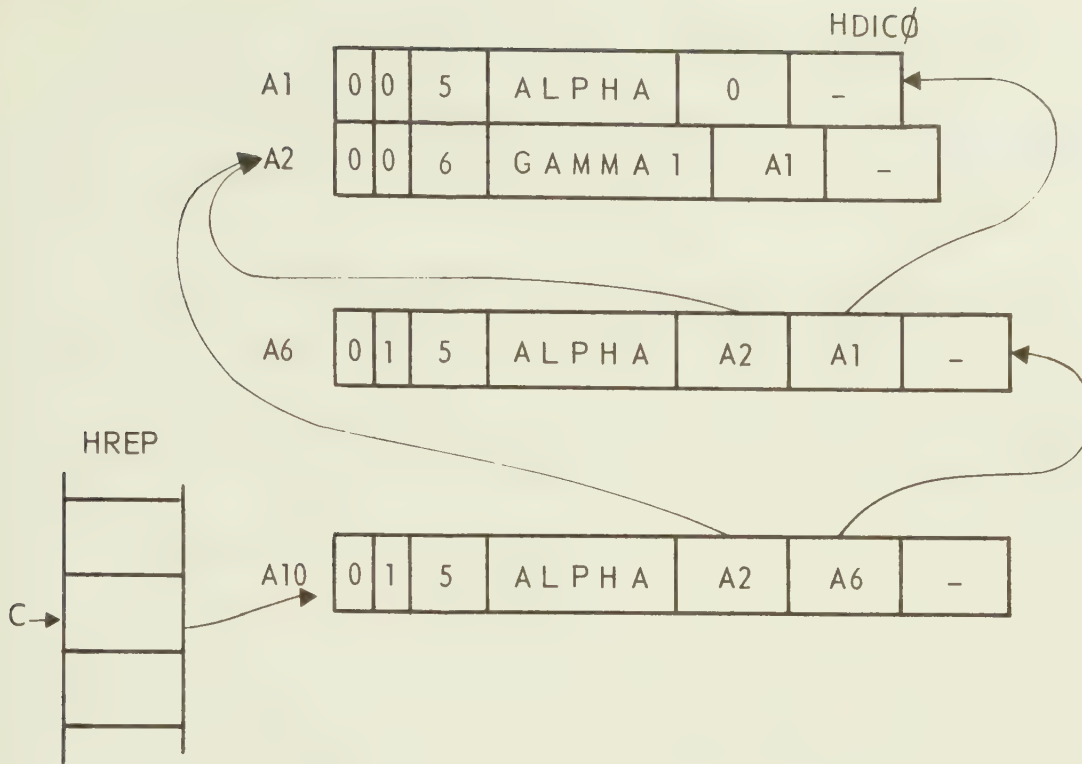
Lors de la rencontre d'un identificateur, l'application d'un algorithme de "Hashing" lui associe un nombre naturel V qui indique son appartenance à la classe V des identificateurs.

V, caractéristique de la classe d'un élément, pointe dans une table auxiliaire HREP, initialisée à zéro, 2 octets qui, lors du rangement, sont garnis avec l'adresse de cet élément dans le dictionnaire.

Pour chaque nouvel élément de classe V rencontré, le contenu de ces deux octets est mis en place dans la zone BLINK de l'élément du HDICØ; ensuite une recherche d'élément identique a lieu; en cas de non identité le processus d'introduction est terminé.

Si un élément identique est trouvé, IPREV prend la valeur 1, une comparaison est faite entre le BLINK provisoire et l'adresse de cet élément, si il y a identité BLINK est déplacé dans BIDLINK, BLINK prend la valeur de la partie BIDLINK de cet élément précédent.

Si la comparaison donne un résultat faux, BLINK est conservé et BIDLINK est l'adresse de l'élément précédent. e.g.



Dans cet exemple on suppose que ALPHA et GAMMA1 appartiennent à la même classe C.

Dans l'élément A1 de HDICØ, BLINK = 0 indique que cet élément est le dernier de la classe (i.e. le premier à être apparu).

HREP comporte autant de fois 2 octets qu'il existe de classes rendues possibles par l'algorithme de Hashing". Le chainage des éléments d'une classe est remontant : i.e, le premier élément accessible d'une classe est toujours le dernier introduit.

La recherche d'un élément dans HDICØ se fait de la façon suivante :

- calcul par l'algorithme de "Hashing" de V ;
- V permet de pointer dans la table des origines des chaînes HREP, l'adresse dans le dictionnaire HDICØ du 1er élément de la classe (dernier introduit) ;
- le parcours de la chaîne achève la recherche.

3.2.6.2 - ALGORITHME DE "HASHING"

L'algorithme utilisé est celui introduit dans le compilateur CØBØL IRIS 50 ; sa mise en oeuvre est simple et il fournit des résultats corrects

L'algorithme de "Hashing" est le suivant :

Tout caractère est représenté par un nombre naturel < 256 . On considère donc un identificateur comme un nombre naturel représenté en base 256^2 (en complétant au besoin cet identificateur par un caractère α à droite). On fait la somme des chiffres modulo 256^2 .

On considère cette représentation comme étant en base 256. On fait la somme des 2 chiffres dont on prend le reste modulo 64.

Ce résultat est multiplié par 2 pour être utilisé comme index dans la table HREP.

La taille de HREP est de 64 demi-mots (128 octets).

A partir de l'analyse des phrases de type 5, les identificateurs ne sont plus entrés dans le dictionnaire, comme les identificateurs apparaissant dans les expressions de longueur, les conditions, les répétitions de 2ème forme et les répétitions de 1ère forme

Cependant ces identificateurs sont générés dans la file intermédiaire avec leur indicatif calculé par l'algorithme de "Hashing", ceci évite de le recalculer ensuite

La phase d'analyse syntaxique retient certaines informations concernant les partitions cataloguées, les manipulations du catalogue des partitions, les macro-instructions du compilateur.

Ces informations permettent à la phase résidente d'opérer les recherches spécifiées (partitions cataloguées, prototypes de macro-instructions, procédures à insérer, etc...) et de constituer une nouvelle file à l'attention de la seconde "passe".

3.3 - PHASE DE VERIFICATION DE PRE-GENERATION

Le premier travail de cette phase, après son appel en mémoire et les différents arrangements nécessaires des informations externes (e.g. partitions cataloguées) est d'opérer une vérification suivie d'une réorganisation de toute la partie "déclaration" de la procédure.

Cette première partie génère d'une part la partie résidente, dans le code, des descriptions et d'autre part un dictionnaire utilisé dans la vérification sémantique et la pré-génération des instructions.

Comme on l'a dit dans le chapitre précédent à propos des appels de procédure (2.8.5), il est nécessaire de conserver toute une série d'informations permettant d'une part de vérifier les arguments lors de l'appel de procédure et d'autre part de générer dans une zone gérée dynamiquement les objets de statut AUXILIAIRE dont la procédure fait usage.

Il existe plusieurs classes d'objets :

- objets simples fondamentaux ;
- index de files ;
- procédures ;
- étiquettes ;
- partitions (sous-partitions) ;
- articles ;
- files.

Selon la classe à laquelle appartient l'élément, et selon le statut de ce dernier, l'information qui subsiste dans le code est différente.

1. Arguments

1.1 - Objets simples fondamentaux

Deux cas sont à distinguer

1. *Longueurs simples*

e.g. NATUREL, ENTIER

Il ne subsiste en mémoire que :

- le type de l'élément ;

- le type de sa longueur, qui peut être :
 - constante
 - non fixée
 - fixée par une expression

2. Longueurs composées

i.e. DECIMAL, REEL

Il ne subsiste que :

- le type de l'élément ;
- les types des deux parties de la longueur.

1.2 - Index de files

Pour un index, il est nécessaire de disposer des informations suivantes ;

- la file à laquelle il est rattaché ;
- l'entrée particulière de cette file à laquelle il est rattaché ;
- les caractéristiques concernant sa valeur ;
- les informations rattachées à la description de la file :
 - bornes de la file,
 - pas de l'entrée

D'ailleurs la valeur de l'index ne peut s'interpréter qu'à l'aide de ces dernières informations.

Pour un index cité parmi les arguments d'une procédure il importe de distinguer deux cas :

- l'index n'est rattaché à aucune file dans la procédure ;
- l'index est rattaché à une file, de statut argument, de la procédure.

Le cas où l'index, de statut argument, est rattaché à une file de statut AUXILIAIRE n'est pas envisagé dans la version actuelle du langage.

Dans chacun des cas la disposition suivante est adoptée :

1. l'index n'est rattaché à aucune file dans la procédure : l'information qui subsiste est une référence permettant de créer le lien à l'exécution avec les caractéristiques de la file dans le programme appelant.

2. l'index est rattaché à une file, de statut argument, de la procédure, dans ce cas il subsiste :

- les références à la file à laquelle il est rattaché.

Les informations :

- bornes de la file ,
- pas de l'entrée,

sont rattachées à chacune des entrées d'une file. En conséquence le qualificatif d'un index n'a pas besoin de comporter ces informations.

1.3 - Procédures

Aucune information particulière ne concerne les procédures, il est juste nécessaire de prévoir une zone permettant de se référer à cet objet (i.e. un nom de procédure sur 8 caractères au maximum).

Les étiquettes sont obligatoirement de statut AUXILIAIRE, elles ne figurent pas dans cette énumération.

1.4 - Partitions

Il s'agit d'une sous-partition d'une partition, dans ce cas il est nécessaire de laisser subsister :

- une référence à la description codée de cette sous-partition ;
- une référence à la description codée de la partition dont l'objet en question est une sous-partition.

1.5 - Articles

Pour un article il est seulement nécessaire de laisser subsister :

- une référence à la description codée de la partition structurant l'article,
- des informations caractérisant la place de cet article et éventuellement des caractéristiques propres aux opérations d'entrée-sortie.

1.6 - Files

Pour une file, les informations suivantes sont nécessaires :

- le nombre d'entrées de la file ;

- pour chacune de ses entrées, le type de la longueur spécifiée :
 - constante
 - variable
 - non mentionnée
 et les informations qui lui sont rattachées (pas en particulier) ;
- une référence à la description codée de la partition structurant les articles de cette file ;
- la place de cette file ;
- dans le cas des fichiers, l'organisation de celui-ci éventuellement.

Dans le cas des arguments, les informations qui subsistent dans le code permettent d'une part de vérifier la concordance des arguments à l'appel de procédure et d'autre part de créer la zone des adresses des arguments et de certains qualificatifs de ces arguments. Comme on l'a mentionné précédemment un appel de procédure est réalisé en transmettant uniquement les adresses des arguments

2. Auxiliaires

Pour les objets de statut AUXILIAIRE, la situation est différente. Les informations qui restent dans le code du programme généré, doivent permettre de générer les objets dans une zone, dite zone des données ; celle-ci est gérée dynamiquement par un sous-programme invoqué à chaque appel de procédure et à chaque fin de procédure (instruction FIN).

2.1 - Objets simples fondamentaux

Pour ces objets la différence vient de l'expression de longueur :

1. Constante (explicite ou longueur implicite)

Il subsiste la valeur de la constante et le type de cadrage (quelconque, demi-mot, mot ou double-mot).

2. Variable. Il subsiste des informations permettant le calcul de la longueur au moment de l'appel de procédure (référence à l'élément intervenant dans l'expression de longueur et type de l'expression) et éventuellement des informations concernant le cadrage.

Pour les objets de nature DECIMAL ou REEL, qui comportent deux expressions de longueur, les informations tiennent, chacune, de l'un des deux types cités ci-dessus.

2.2 - Index de files

Deux cas se présentent :

- l'index est rattaché à une file de statut argument ;
- l'index est rattaché à une file de statut AUXILIAIRE

dans chacun de ces deux cas, l'information est différente.

1er Cas

Il subsiste d'une part le type de la zone qui contient la valeur de l'index (demi-mot ou mot) et le cadrage associé, et d'autre part une référence à la zone des caractéristiques de la file parmi les informations concernant les arguments.

2ème Cas

Il subsiste d'une part le type de la zone qui contient la valeur de l'index (demi-mot ou mot) et le cadrage associé, et d'autre part une référence aux caractéristiques de la file parmi les informations concernant les auxiliaires.

2.3 - Procédures

Une variable de statut AUXILIAIRE et de nature ~~PROCEDURE~~ doit faire obligatoirement l'objet d'une définition en phrase de type 5, cet objet devient une constante et appartient donc au programme généré.

La situation est différente pour les objets de nature ~~PROCEDURE~~ cités dans une partition.

2.4 - Etiquettes

Ce type d'objet n'apparaît pas dans les informations à ce niveau.

2.5 - Partition

Il s'agit d'une sous-partition d'une partition, dans ce cas il est nécessaire de laisser subsister :

- la description codée de cette sous-partition ;
- la référence à la description codée de la partition dont l'objet en question est une sous-partition.

2.6 - Articles

Pour un article, il est nécessaire de laisser subsister :

- une référence à la description codée de la partition structurant l'article ;
- une information caractérisant la place de cet article et éventuellement des caractéristiques propres aux opérations d'entrée-sortie.

2.7 - Files

Pour une file, il est nécessaire de laisser subsister :

- le nombre d'entrées de la file,
- pour chacune de ses entrées, la longueur spécifiée :

1 Constante

Il subsiste la valeur de la constante.

2. Variable

Il subsiste des informations permettant le calcul de la longueur au moment de l'appel de procédure.

et les informations attachées (bornes, pas de l'entrée).

- une référence à la description codée de la partition structurant les articles de cette file.
- la place de cette file ;
- pour le cas des fichiers, l'organisation de celui-ci éventuellement

Le cas des partitions est un peu particulier ; comme on l'a dit précédemment, lors de l'appel d'une procédure les objets de statut AUXILIAIRE et de support MEMOIRE sont générés automatiquement, il est donc nécessaire de conserver une forme codée de la partition structurant un article ou une file.

De même pour les articles et les files figurant dans un appel de procédure, il est nécessaire de vérifier la correspondance entre les structures des éléments mis en regard par l'intermédiaire de la séquence d'appel.

La description codée d'une partition reprend donc les descriptions explicitées ci-dessus pour les objets simples fondamentaux en particulier, des informations analogues pour les éléments simples spécifiques des partitions (RIEN, ZERO, etc...).

Dans le cas des éléments de nature PROCEDURE ou ETIQUETTE, la structure est un peu différente :

- ces objets doivent faire l'objet d'une définition (phrase de type 5).

la description codée de la partition contient l'adresse de la zone associée à la définition.

Dans le cas d'éléments répétés et/ ou conditionnels des informations complémentaires sont adjointes aux descriptions de ces éléments.

La méthode adoptée pour la vérification des déclarations est inspirée de celle décrite par KNUTH (18), et utilise des structures multichainées, ces structures sont toujours associées à une technique de "Hashing".

Cette partie de la seconde "passe" permet de créer le dictionnaire des références croisées, ainsi que la géographie (MAP) des objets du traitement.

Dans le cas le plus général, la structure d'un élément du dictionnaire de vérification est la suivante :

IND	T1	T2	PREVIOUS	FATHER	PARAL	LG1	NAME
-----	----	----	----------	--------	-------	-----	------



avec :

IND	ensemble d'indicateurs
T1	Type de l'élément (simple, répété 1ère forme, etc...)
T2	Nature de l'élément
PREVIOUS	Adresse du précédent élément de même nom
FATHER	Adresse de la partition (élémentaire) à laquelle appartient l'élément
PARAL	Adresse de la zone contenant les informations concernant les éléments mis en parallèle avec l'élément considéré.
LG1	Longueur du nom de l'élément (nombre de caractères)
NAME	Nom de l'élément
SØN	Adresse de la description de partition, désignée par NAME, (cas où l'élément de partition est une partition).
BRØTHER	Adresse de l'élément suivant dans la description de partition.

LG2 Longueur de la zone

ZONE Zone contenant des informations complémentaires (répétition, condition etc...).

Les algorithmes de vérifications sont écrits et en cours de programmation.

Une fois opérée cette vérification des descriptions, les instructions sont analysées et un précode est généré.

C'est à ce niveau qu'interviennent les macro -instructions ayant une influence sur le programme résultant.

3.4 - PHASE DE GENERATION

Cette phase n'est pas définie avec précision pour l'instant, cependant elle ne semble pas poser de gros problèmes. Elle travaille à partir du pré-code généré par la seconde passe, et donne :

- si spécifié dans les paramètres de la carte de contrôle invoquant le compilateur, la liste en langage type assembleur des ordres générés (option $L\emptyset$) ;
- un module objet standard compatible avec les spécifications de l'éditeur de liens.

La création de cette 3ème "passe" correspond à trois préoccupations :

- réduire l'occupation mémoire du compilateur ;
- simplifier la seconde "passe" de vérification en la libérant du problème de génération ;
- essais d'optimisation du code généré

3.5 - CONCLUSION - EXTENSION CONVERSATIONNELLE

La division logique du compilateur en trois "passes" correspondant à trois phases s'exécutant en "overlay" avec une partie résidente a été motivée par les considérations suivantes :

- diminution de la place mémoire occupée par le compilateur ;
- structuration des différentes parties du compilateur du fait de la définition de formats pour les informations intermédiaires ;
- génération d'un code machine relativement "bon" et possibilité d'optimisation ;
- prise en compte des modifications ;
- extensions du compilateur.

Ces deux derniers points méritent quelques commentaires.

En effet au moment où la réalisation du compilateur a débuté, le langage était en phase de définition, il importait de prendre en compte facilement les modifications. De même, lors des premières applications du langage, il peut s'avérer que certaines facilités soient inutiles ou bien au contraire à introduire dans le langage ; dans ce cas la possibilité de modifications aisées est très précieuse.

L'extension du compilateur peut se faire, essentiellement, dans deux voies :

- conversationnel ;
- macro -instructions proposées par l'utilisateur.

L'extension vers la compilation du langage en mode conversationnel, est relativement aisée avec l'organisation adoptée.

La forme conversationnelle du langage n'est pas définie avec précision, quoiqu'il en soit, les modifications éventuelles n'auront que peu d'influence sur la partie analyse syntaxique du fait de l'introducteur de syntaxe (programme CHARADE).

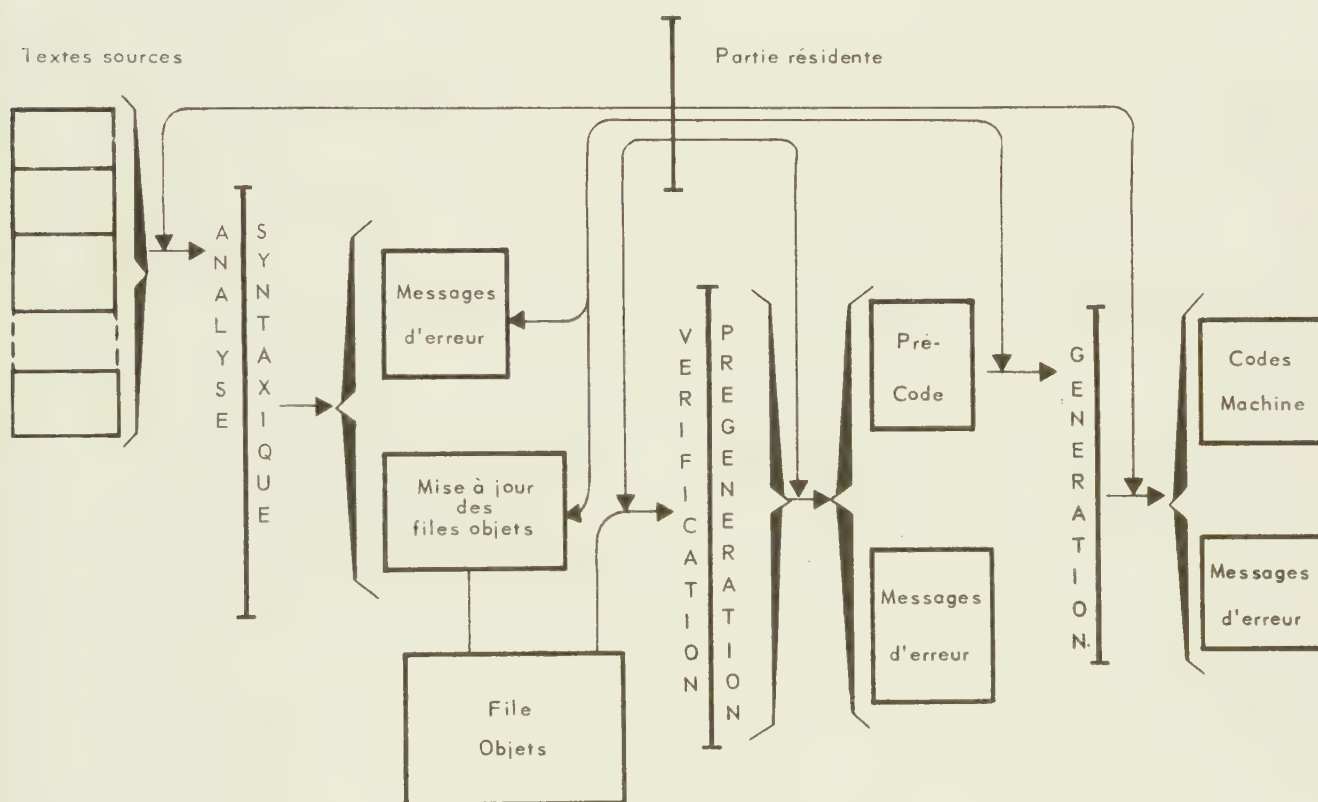
Le traitement envisagé est le suivant :

- analyse syntaxique "on-line" ;
- vérification et génération du type "batch processing" ;
- exécution du type classique après édition des liens et changement, l'utilisateur ayant la possibilité de demander, en cours d'exécution la valeur de certains paramètres, mais pas celle de modifier directement le programme ; toute modification entraîne une remise

à jour de la file objet servant de donnée à la 2ème "passe" du compilateur et donc une recompilation à partir de ce niveau.

La solution "incrémentielle" n'a pas été envisagée car elle entraîne plus de modifications de la partie vérification-génération du compilateur ; elle risque, en outre, de conduire à des temps d'exécution prohibitifs vu le grand nombre de vérifications qu'il serait nécessaire d'opérer au cours de l'interprétation des pseudos-instructions.

Avec la solution adoptée, il n'est nécessaire de réécrire que la partie résidente du compilateur, les trois phrases ne subissant que des modifications mineures. L'organisation est la suivante :



La partie résidente, en relation avec le système d'exploitation, fournit à la phase d'analyse syntaxique l'adresse d'une zone contenant le texte source et les adresses de deux zones destinées à recevoir les messages d'erreur éventuels d'une part et d'autre part les mises à jour de file objet. La partie résidente assure l'impression des messages d'erreur et/ ou la mise à jour de la file objet, elle fournit à la seconde "passe" les adresses des deux zones, file objet entrée et messages d'erreur. Après l'exécution de cette phrase, la partie résidente imprime les messages d'erreur et/ ou lance l'exécution de la phrase de génération. Après quoi,

elle imprime les messages ou demande l'édition et l'exécution du code généré.

Cette extension ne prétend pas être une solution idéale mais elle permet de réaliser, à peu de frais, un compilateur conversationnel.

H. KATZAN a publié, depuis, un article exposant en détail des idées similaires (16).

4 — CONCLUSION

Cette étude a permis, d'une part l'évolution du langage proposé (22) et d'autre part la définition de la structure du compilateur ainsi que le début de sa réalisation

Tout au long de ce travail, les différents choix ont été faits en fonction de critères simples :

- souci d'une définition cohérente du langage ;
- possibilités d'adaptation aux applications de gestion, objet du langage ;
- facilités de traitement sur l'ordinateur choisi.

Cette façon de procéder, à partir d'un langage complet et logiquement défini, l'ATF (21), laisse prévoir de nouveaux langages adaptés à des classes de problèmes

La mise en oeuvre de la méthode d'analyse syntaxique descendante s'est avérée très souple d'emploi, la version française et la version anglaise du langage étant très semblables du point de vue du compilateur

Le codage du compilateur en langage d'assemblage permet de chiffrer avec exactitude le coût des différentes facilités du langage, et de plus les observations accumulées au cours de la réalisation peuvent permettre de faire évoluer la structure des machines vers des solutions plus adoptées au traitement des problèmes tant de compilation que d'exécution

BIBLIOGRAPHIE

- 1 - FELDMAN J. and GRIES D. Translator Writing Systems Com. ACM February 1968
- 2 - GLASS Robert L An elementary discussion of compiler/Interpreter Writing
Computing Surveys March 1969
- 3 - IBM System/360 Principles of Operation A22-6821-6
- 4 - IBM System/360 Assembler Language C28-6514-3
- 5 - IBM System/360 Durée des instructions 360 A11-1210-0
- 6 - IBM System/360 Operating System

Job Control Language Charts C28-6632-3
- 7 - IBM System/360 Operating System
Concepts and Facilities C28-6535-2
- 8 - IBM System/360 Operating System
Supervisor and Data Management Services
C28-6646-1
- 9 - IBM System/360 Operating System
Storage Estimates C28-6651-6
- 10 - IBM System/360 Operating System
System Control Blocks C28-6628-1
- 11 - IBM System/360 Operating System
Supervisor and Data Management Macro-Instructions
C28-6647-2
- 12 - IBM System/360 Operating System
System Programmer's Guide C28-6550-3
- 13 - IBM System/360 Operating System
Programmer's Guide to Debugging
C28-6670-0
- 14 - JAMARD P Technique de Hashing
Document CII Décembre 1968

ANNEXES

ANNEXE 1

VERSION ANGLAISE DU LANGAGE

Cette version se distingue de la version présentée ici en langue française par le seul changement des mots réservés du langage. La liste suivante donne les mots réservés de la version anglaise, l'équivalent dans la version française est mentionné entre parenthèses

ABS	
ADLIB	
AFTER	(APRES)
AND	(ET)
AUXILIARY	(AUXILIAIRE)
BACK	(AR)
BEFØRE	(AVANT)
BLANK	(BLANC)
BØØLEAN	(BØØLEEN)
BØUNDEDTØ	(LIMITEA)
BY	(BACK ou FØRWARD < index > BY)
BYTE	(MØT)
CATAL	
CLEAR	(EFFACER)
CØNC	(PUIS)
CØRE	(MEMØIRE)
DATUM	(DØNNEE)
DECIMAL	
EMPTY	(VIDE)
END	(FIN)
ERRØR	(ERREUR)
FALSE	(FAUX)
FILE	
FIRST	(PREMIER)
FØRWARD	(AV)

FRØM	(DE- JUSQUA -)
GREATER	(SUPER)
GØ TØ	(VERS)
ICARD	(ECARTE)
IF	(SI)
INDEX	
INTEGER	(ENTIER)
ITEM	(ARTICLE)
IPAPERTAPE	(ERUBAN)
IMMEDIATPROCESSING	(TRAITEMENT IMMEDIAT)
KEYBØARD	(CLAVIER)
LABEL	(ETIQUETTE)
LAST	(DERNIER)
LENGTH	(LØNG)
LESS	(INFER)
LISTING	(IMPRIME)
MACRØ	
MIXT	(MIXTE)
MØD	
NATURAL	(NATUREL)
NEAR	(PRØCHE)
NØ	(NØN)
NØTHING	(RIEN)
ØCARD	(SCARTE)
ØF	(DE)
ØPAPERTAPE	(SRUBAN)
ØR	(ØU)
ØRD	
ØRDER	(ØRDRE)
PAPER	(PAPIER)
PRE	
PRØCEDURE	
Q (Q	(G (G)
Q) Q	(G) G)
REAL	(REEL)
REMAINDER	(RESTE)

RESULT	(RESULTAT)
RØUNDEDTØ	(ARRØNDIA)
SIGN	(SIGNE)
STØCK	
STRUCTURE	(PARTITIØN)
SYNCHRØ	
TAPE	(BANDE)
TEN	(AINE)
TENTH	(IEME)
TRUE	(VRAI)
UNIT	(UNITE)
UNTIL	(JUSQUA)
ZERØ	

ANNEXE 2

GRAMMAIRE DU LANGAGE

Cette annexe contient une version intermédiaire de la grammaire du langage sous la forme utilisée par le programme introducteur de syntaxe ainsi que des parties des listes générées par ce programme.

GRAMMAIRE CODELAB -VERSION FRANCAISE-

AXIOME

AXIOME =

ad DCO aSa(£a)a(ar TOIM aSa(ar T1 a)aSa(ar T2IM a)a|ar TO aSa(ar
T1 a)aSa(ar T2 a)a)
ar T3 aSa(ar T4 a)aSa(ar T5 a)ar T6 ar T7 af

PHRASE DE TYPE 0 (DEBUT DU PROGRAMME)

TOIM =adTRAITIM aSa(£a);aJERRPTVIR aI FINTOIM af
TO = arIDENT aSa(£a);aJERRPTVIR aI FINTO af
IDENT =aIIDENTIF af

PHRASES DE TYPE 1 (REFERENCES)

T1 =aDDGENTY1

aSa(£a)arREFE aDDAD1 aSa(£a)aSa(£a) ar IDENT aJERR2
aDDAD1 aSa(£a)a);aJERRPTVIR aI FINT1 af

REFE =aIREF af

PHRASES DE TYPE 2 (DESCRIPTIONS)

T2 =aDDGENT2 aSa(£a)a(a(DaDDDDONNEE aSa(£a) ar ARG aSa(£a)a)
a|a(MaDDDMIXTE aSa(£a) ar ARG aSa(£a)a)
a|a(RaDDRESULTA aSa(£a) ar ARG aSa(£a)a)
a|a(AaDDAUXILI aSa(£a) ar ARGa aSa(£a)a)aI FINT2 af

ARG =ar PER aSa(£a)a(EN aJ ERRDECRI ar T2ENT a|ar ARG1 a)af
ARG1 =A ar T2ART a|B ar T2BOOL a|D ar T2DEC a|F ar T2FILE a|I ar T2IND
a|M ar T2MOT a|N ar T2NAT a|O ar T2ORD a|P(a ar T2PART a|R
ar T2PROC a)a|R aJ ERRDECRI ar T2REEL af

PER =& a JERRSEPAR af

T2IM = aDDGENT2

aSa(£a) a(AaDDAUXILI1aSa(£a)arARG aSa(£a)a)aI FINT2IM af

ARGa = ar PER aSa(£a)a(Ea(N ar T2ENT a|T aJ ERRDECRI ar T2ETIQ a) a|
ar ARG1 a)af

T2ETIQ =

ad DETIQUET aSa(£a)ar PER aSa(£a)ar IDENT
aSa(£a)a(&aSa(£a)a(&aSa(£a)a(&aSa(£a)a(&aSa(£a)a(&aSa(£a)a
a(&aSa(£a)a(M ad DMEMOIR1 a|av)a|av)a|av)a|av)a|av)a
a|av)a|av)a); aJ ERRPTVIR af

T2ART =aDDARTICLE aSa(£a)ar PER aSa(£a)ar IDENT aSa(£a)ar PER
aSa(£a)ar PER aSa(£a)ar IDENT aSa(£a)ar PER aSa(£a)
ar PLACEPAR aSa(£a)a(&aSa(£a)a(&aSa(£a)a(&aSa(£a)a(ar PLACE
aSa(£a)a|aDDGENMEM a)a|aDDGENMEM a)a|aDDGENMEM a)
;aJ ERRPTVIR aI VERIFPLA af

PLACEPAR =a(Cad DCATAL a)a|ad DGENVIDE af

PLACE =a(Bad DBANDE a|C ad DCLAVIER a|Ea(C ad DCARTE a|R ad DERUBAN a)
a|I ad DIMPRIME a|M ad DMEMOIRE a|P ad DPAPIER a|Sa(C ad DCARTS
a|R ad DSRUBAN a|T ad DSTOCK a)a)af

T2BOOL =aDDBOOLEEN aSa(£a)ar PER aSa(£a)ar IDENT aSa(£a)ar PER aSa(£a)
ar LONG1 aSa(£a)ar PLADLIB1 af

LONG1 =a(ar LONGNAT a|ar EXPR a|ar LONGEXPQ a|ad DGENRIEN a)
aI VERIFLG af

LONGNAT =ad DGENAT aI NNATUREL af

LONGEXPQ = ad DGENVAR1 ar EXPQUA2 af

EXPQUA2 = ad DGENIET ar IDENQUA2 ad DGENPLU1 aSa(£a) aSa(DE£ aSa(£a)

@R IDENQUAL @S@(@) @I FINEXQUA @F
 NBNAT =@I NNATUREL @F
 EXPR =
 @ (ORD@ @D DGENVAR3 @|PRE@ @D DGENVAR4 @) @S@(@)
 @R EXPQUA2 @F
 PLADLIB1=@(&@S@(@)@(&@S@(@)@(&@S@(@)@(&@S@(@)@(&@S@(@)@
 M @D DMEMOIR1 @|@V@)@|@V@)@|@V@)@|@V@)@|@V@)@|@V@)@S@(@); @J
 ERRPTVIR @F
 T2DEC =
 @(@D DDECIMAL @S@(@)@R PER @S@(@)@R IDENT @S@(@)@R PER
 @S@(@)@)@(@R LONGD . @J ERRPTDEC @R LONGD1 @|@D DGENRIEN @(
 @R LONGD1 @|@D DGENRIEN @)@)@D DVERIDEC @S@(@) @R PLADLIB1 @F
 T2ENT =@DDTIER @S@(@)@R PER @S@(@)@R IDENT @S@(@)@R PER @S@(@)
 @R LONG1 @S@(@)@R PLADLIB1 @F
 LONGD = @R LONGNAT @|@R EXPR @|@R LONGEXPQ @F
 LONGD1 =@R LONGD @|@D DGENRIEN @F
 T2FILE =@(@D DFILE @S@(@)@R PER @S@(@)@R IDENT @S@(@)
 @R PER @S@(@)@R LONGFILE @S@(@)@R PER @S@(@)@R IDENT @S@(@)
 @R PER @S@(@)@R PLACEPAR @S@(@)@R PER @S@(@)@R INDEX @S@(@)@
 @D VERINDEX @(&@S@(@)@(@R PLACE @S@(@)@|@D DGENMEM @)@|
 @D DGENMEM @) ; @J ERRPTVIR @I VERIFPLA @F
 LONGFILE=@D DRAZNE @(@@R EXPRFILE @|@R LONG1 @)@D DNE1 @)
 @S@(@)@S@(@, @S@(@)@(@R EXPRFILE @|@R LONG1 @)@D DNE1
 @S@(@)@) @F
 EXPRFILE = LONG@ @S@(@) @D DGENLONG @R IDENT @D DRAZNID @S@(@)
 @ (@S@(@)@R PARENT @|@V@)@F
 PARENT = @(@R LONGEXPQ @| . @J ERRLONG @D DGENPOIN @)@D DNI1 @S@(@ @S
 @(@) , @D DNI1 @S@(@) @(@R LONGEXPQ @| . @J ERRLONG @D DGENPOIN @)
 @)@S@(@)) @J ERRPAREN @F
 INDEX =@DDRAZLT @D DC1 @S@(@)@R PER @S@(@)@R IDENTR @S@(@)
 @S@(@R 8PER @S@(@)@R NBNAT @D GENINDEX @S@(@)@R PER
 @S@(@) @R IDENTR @) @I VERINE @F
 8PER =
 &@S@(@)&@S@(@)@D D6PERLUE @F
 IDENTR = @D DRAZNI @(@R IDENT @D DNIPLUS1 @S@(@) @S@(@ , @D
 DNIPLUS1 @S@(@) @R IDENT @S@(@)@|@V@)@F
 T2IND = @D DINDEX @S@(@)@R PER @S@(@)@R IDENT @S@(@)@R PER @S@(@)
 @R LONG1 @S@(@)@R PLADLIB1 @F
 T2MOT =@DDMOT @S@(@)@R PER @S@(@)@R IDENT @S@(@)@R PER
 @S@(@)@R LONG1 @S@(@)@R PLADLIB1 @F
 T2NAT =@D DNATUREL @S@(@)@R PER @S@(@)@R IDENT @S@(@)@R PER
 @S@(@)@R LONG1 @S@(@)@R PLADLIB1 @F
 T2ORD =@D DORDRE @S@(@)@R PER @S@(@)@R IDENT @S@(@)@(&@S@(@)
 @R PLADLIB1 @| ; @J ERRPTVIR @D DGENMEM @) @F
 T2PART = @D DPARTITI @S@(@)@R PER @S@(@)@R IDENT @S@(@)@(& @S@(@)
 @(&@S@(@)@(@R IDENT @S@(@)@(&@S@(@)@R PLACEPAR @|@D DGENVIDE
 @)@|@D GENLGO @(&@|@V@)@|@D GENLGO @)@|@D GENLGO @) @S@(@)
 ; @J ERRPTVIR @F
 T2PROC =@D DPROCEDU @S@(@)@R PER @S@(@)@R IDENT @S@(@)@(&
 @S@(@)@(&@S@(@)@(&@S@(@)@(&@S@(@)@(&@S@(@)@(&@S@(@)
 M@D DMEMOIR1 @|@V@)@|@V@)@|@V@)@|@V@)@|@V@)@|@V@)@S@(@)
 ; @J ERRPTVIR @F
 T2REEL = @D DREEL @S@(@)@R PER @S@(@)@R IDENT @S@(@)@R PER
 @S@(@)@(@@R LONGD , @J ERRVIR @R LONGD1 @)@|@D DGENRIEN @ (,
 @R LONGD1 @|@D DGENRIEN @)
 @D VERLGREL @S@(@)@R PLADLIB1 @F

PHRASES DE TYPE 3 (PARTITIONS)

T3 = @S@(@R T3C @S@(@)@(& @S@(@) @(@R IDENT @S@(@)@RT4A @|
 @S@(@& @S@(@)@R T3A @S@(@)@)@|@V@) ; @J ERRPTVIR
 @D DFINPART @) @I FINT3 @F

T3A = @D DTEGALO @(&@S@(@@)@R T3012 @|@R CONDTIO @J ERRPARTI @D DTEGAL2
 @S@(@@)@R PER @S@(@@)@R T3012 @|@F
 T3012 =@(& @S@(@@)@R T3020 @|@R DRAPEAU @J ERRPARTI @S@(@@)@RSUITE1 @|@F
 T3020 =@(& @|@R NBFOIS @J ERRPARTI @S@(@@)@R PER @|@S@(@@)@R SUITE2 @F
 DRAPEAU =@(' @D DTEGAL1 @R CHAINE @|@D DTEGALO @R EXPQUA @|@S@(@@)@(@ ,
 @S@(@@)@R LONG1 @|@D GENRIEN @|@F
 CHAINE =@I CHAINE @F
 SUITE1 =@D DTPLUS3 @R PER @S@(@@)@(& @|@R NBFOIS @J ERRPARTI @S@(@@)
 @R PER @|
 @S@(@@) M @D DMOT @S@(@@)@R PER @S@(@@)@R LONG1 @D VERLG1
 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@I MJLTP @F
 T3C =@D DGENT3 @S@(@@)@R T3B1 @F
 T3B1 = @R IDENT @S@(@@) @R PER @S@(@@) @(@R T3A1 @|@R T3A @) @F
 T3A1 =& @S@(@@) & @S@(@@)& @S@(@@) SYNCHRO @D GENSYNC @S@(@@)@R PER
 @S@(@@) @ (1 @D GENSY1 @| 2 @D GENSY2 @| 4 @D GENSY4 @| 8 @D GENSY8
 @|@S@(@@)@R PER @F
 T3A2 =& @S@(@@) & @S@(@@) & @S@(@@) @ (SYNCHRO @D GENSYNC1 @S@(@@)@R PER
 @S@(@@)@ (1 @D GENSY1 @|2 @D GENSY2 @|4 @D GENSY4 @|8 @D GENSY8 @)
 @|@R RIENSYN @J ERRSYNCH @|@S@(@@)@R PER @F
 RIENSYN = RIEN @S@(@@) @D GENRIEN @R PER @F
 NBFOIS =@D DTPLUS4 @D NBEGALO @(@R NBNAT @D NBEGAL1 @|@R EXPQUA @|@F
 SUITE2 =@ (A @R SUITE21 @|B @ (L @R SUITE22 @|O @R SUITE23 @|@|
 D @R SUITE24 @|E @ (N @R SUITE25 @|T @R SUITE26 @|@|I @R
 SUITE27 @|M @R SUITE28 @|N @R SUITE29 @|P @ (A @R SUITE30
 @|R @R SUITE31 @|@|R @ (E @R SUITE32 @|I @R SUITE33 @|@|Z
 @J ERRNAT @R SUITE34 @|@F
 SUITE21 =@(@D DADLIB @S@(@@)@R PER @S@(@@)@R PER @S@(@@)@(@R
 IDENT @|@D GENLGO @|@)@I MAJLTPRI @F
 SUITE22 =@(@D DBLANC @S@(@@)@R PER @S@(@@)@R LONG1 @D VERLG1
 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE23 =@(@D DBOLEEN @S@(@@)@R PER @S@(@@)@R LONG1 @D VERLG1
 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE24 =@(@D DDECIMAL @S@(@@)@R PER @S@(@@)@(@@R LONGD . @J
 ERRPTDEC @R LONGD1 @|@|@D DGENRIEN @ (. @R LONGD1 @|@D DGENRIEN @|@)
 @D DVERIDEC
 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE25 =@(@D DTIER @S@(@@)@R PER @S@(@@)@R LONG1 @D VERLG1
 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE26 =@(@D DETIQUET@S@(@@)@R PER @S@(@@)@RLONG1 @S@(@@)@R PER @S@(@@)
 @(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE27 = @(@D DINDEX @S@(@@)@R PER @S@(@@)@R LONG1
 @D VERLG1 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D
 GENLGO @|@)@I MJLTP @F
 SUITE28 =@(@D DMOT @S@(@@)@R PER @S@(@@)@R LONG1 @D VERLG1
 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE29 =@(@D DNATUREL @S@(@@)@R PER @S@(@@)@R LONG1 @D
 VERLG1 @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE30 =@(@DDPARTITI@S@(@@)@RPER @S@(@@)@ (C@DDCATAL @|@DDGENVIDE@)@S@(@@)
 @S@(@@)
 @|@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE31 = @(@D DPROCEU @S@(@@)@RPER @S@(@@)@R LONG1 @S@(@@)@R PER @S@(@@)
 @|@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 SUITE32 =@(@D DREL @S@(@@)@R PER @S@(@@)@(@@R LONGD , @J ERRVIR
 @R LONGD1 @|@|@D DGENRIEN @ (, @R LONGD1 @|@D DGENRIEN @|@)
 @D VERLGREL @S@(@@)@R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)
 @I MJLTP @F
 SUITE33 =@(@D DRIEN @S@(@@)@R PER @S@(@@)@R PER @|@I MJLTP @F
 SUITE34 =@(@D DZERO @S@(@@)@R PER @S@(@@)@R LONG1 @D VERLG1 @S@(@@)
 @R PER @S@(@@)@(@R IDENT @|@D GENLGO @|@)@I MJLTP @F
 CONDTIO = @D TCEGALO @(@R EXPQUA @(@R SYMBCOMP @D GENNN @S@(@@)@(
 @R NB@S@(@@) @S@ (| @S@(@@)@R NB @J ERR7 @D NNPLUS1 @S@(@@)@(
 @|@(@R EXPQUA @D TCEGAL3 @S@(@@)@S@ (| @S@(@@)@R EXPQUA @J ERR7


```

AD NNPLUS1 )))))) TCEGAL2 )))) F
SYMBCOMP = ( (= AD GENSI )) (< AD GENS2 )) > AD GENS4 )) (= AD AJOUTER1
))))) # AD GENS6 )) F
NB =AR NB1 AS((E AS((AR EXPO AJ ERREXPO ))))) NBBOOL1 AF
NB1 =AI NB1 AF
EXPO =AI EXPOSANT AF
EXPQUA = AD GENLGT AR IDENQUAL AS((AS(DEAS((AR IDENQUAL AS((
))) AI FINEXQUA AF
IDENQUAL =AD GENRIEN AD GENLGI ARIDENT1 AS((((AD GENO1 AS(( AR S1
)) L AD DLIMITEA AS((AR EXPQUA )) AI FIDENT )) AI FINIDQUA AF
EXPQUAL =AD GENLGT AR IDENQUAL AS((AS(DEAS((AR IDENQUAL ))
AR NBNAT AS(( AI MAJTYTE )) AS(( AI FINEXQUA AF
IDENQUA2 = AU DGENRIEN AD GENLGI ARIDENT1 AS((((AD GENO1 AS((
AR S1 )) L AD DLIMITEA AS(( AR EXPQUA )) AI FINIDQUA )) AR IDENVRAI
AJ IDENSEUL AR IDENQUAL )) F
IDENVRAI = DE E AS(( AI FIDENT AF
S1 = ((AR NBNAT AS(( )) AS((AR S21 )) ' AD GENO2 AR S22 )) .
AD TEXPLUS1 AS(((( , AD TEXPLUS1 AS(( AR INTERFIL )) ))
)) ((AR S24 AS((AR EXPARFIL E AS(( J AD DJUSQUA E AJ ERSEPMOC
AS((AR EXPARFIL AS(((( , AD TEXPLUS1 AS(( AR INTERFIL
AS(( ))) )) )) AD GENO3 AR EXPQUA AR S4 )) F
S21 = (( ( AS(((' AD GENO2 AR S22 )) AR S25 )) AD GENO3 AR EXPQUA
AR S23 )) )) )) F
S22 =AR CHAINE AS(( )) AS(( ( AD GENO8 AS((AR CONDITIO AS((
)) )) )) F
S23 = (( )) AS(( ( AD GENO8 AS((AR CONDITIO AS(( )) )) )) ))
AR TRANCOND )) F
S24 = DE E AD GENO6 AF
S25 = AD GENO1 ARNBNAT AS(( )) AS(( ( AD GENO8 AS((
AR CONDITIO AS(( )) )) )) )) F
TRANCOND = AD TRANCOND AR SYMBCOMP AD GENNN AS((((AR NB AS((AS(
)) AS((AR NB AJ ERR7 AD NNPLUS1 AS(( )) ((AR EXPQUA AD TCEGAL3
AS((AS( | AS((AR EXPQUA AJ ERR7 AD NNPLUS1 )) )) )) AJ
ERRPAREN AF
S4 = (( )) AS((AR S21 )) )) (( , AS((AR INTERFIL AS(( )) )) AR
TRANCOND AF
INTERFIL =(( ((AD GENO4 )) AR DEJUSQUA )) AS(( )) AR EXPQUA )) AS(( , AS((
)) )) ((AD GENO4 )) AR DEJUSQUA )) AS(( )) AR EXPQUA )) )) AJ
ERRPAREN AF
DEJUSQUA = DE E AD GENO6 AS((AR EXPARFIL J AD DJUSQUA E AJ ERSEPMOC
AS((AR EXPARFIL AF
EXPARFIL =((AR PREMDERN )) AS(( )) AD GENO1 AR EXPQUA AF
PREMDERN = P AD PREMIER1 E AD GENO2 )) D AD DERNIER1 E AD GENO3 AF
C
C PHRASES DE TYPE 4 (PARTITIONS EN PARALLELE)
C
T4A =AD DGENT41 AR PER AS((((AD TESTSYN AR T3A2 )) AR T3A )) AS((
AS(( & AS(( AR PER
AS(( AR T3A AS(( AR PER AS(( AR PER AS(( AR T3A AS((
)) ; AJ ERRPTVIR AD FINPART2 AD STACK AI FINT4A AF
T4 = AD DGENT4 AS((AR T3B1 AS((AR PER AS((AD DGENPAR2 AR IDENT
AS(( AR PER
AS(( ((ADTESTSYN AR T3A2 )) AR T3A )) AS((AS(&AS((ARPER AS((
)) AR T3A AS(( AR PER AS(( AR PER AS(( AR T3A AS(( ));
AJ ERRPTVIR AS((AD FINPART2 AI FINT4 AF
C
C PHRASES DE TYPE 5 (DEFINITIONS)
C
T5 = ((ADGENT5 AREXPQUA AD GENCB AD GENCV AS((AR PER AS((
AR VALEUR AJ ERR8
;AJ ERRPTVIR AS(( )) AI FINT5 AF
VALEUR = AR VALEUR1 AD CBPLUS1 AD CVPLUS1 AS(( , AS((AR VALEUR1

```


AD CVPLUS1 ASa(£a)aSa(| AD CBPLUS1 AD GENCV ASa(£a)AR VALEUR1
AD CVPLUS1 ASa(£a)aSa(, ASa(£a)AR VALEUR1 AD CVPLUS1 ASa(£a)a)aF
VALEUR1 =ADTVEGALO a(AR PSEUDO AD TVEGAL2 a| ' AD TVEGAL1 AR CHAINE
a|AR NB a|AR IDENT1 AD TVEGAL3 a)aF

C

C TERMES

C

PSEUDO = AR NBNAT £aSa(£a) Aa(V AD DVDERN a|P AD DPPREM a) aF
TERME = AD GENLGO a(AR TER6 a|AR NB ASa(£a)a(AR SYMBCOMPASa(£a)a(AR NB
a|AR EXPQUA a)a|AR TER1 aJ ERRTERMEa)a|+ AD GENPLUS ASa(£a) AR
EXPQUA a(AR ARRONDIA a|aVa)a| _ AD GENMOINS ASa(£a)AR TER4 a|AR
TER2 a|AR TER7 a|AR EXPQUA aJ ERRTERME a(AR SYMBCOMP ASa(£a)a(AR
NB a|AR EXPQUA aJ ERRTERME a)a|AR TER1 aJ ERRTERME a)a)aF
TER1 = a(a(a(+ AD GENPLUS a|- AD GENMOINS a) ASa(£a) AR IDENT2 ASa(£a)
a(AR ARRONDIA a|aVa)a)a|a(/ AD GENDIV ASa(£a) AR IDENT2 ASa(£a)
a(R AD DRESTE ASa(£a) AR EXPQUA a| AR ARRONDIA a|aVa)a)a|a(*a(*
AD GENEXPON ASa(£a) AR IDENT2 a| AD GENMULT ASa(£a) AR IDENT2
ASa(£a)a(AR ARRONDIA a|aVa)a)a)a|a(M AD DMOD ASa(£a) AR IDENT3
ASa(£a)a(AR ARRONDIA a|aVa)a|P AD DPUIS £aSa(£a) AR IDENT4 a|A
a(V AD DVDERN a|P AD DPPREM a) a| OU £ AD GENOU ASa(£a) AR IDENT5
a| ET £ AD GENET ASa(£a) AR IDENT5 a| AR ARRONDIA a)a|aVa) aF
CHAINE a|AR NB a|AR EXPQUA a)aF
TER21 = a(A AD DBSCONV a|B AD BOOCONV a|D AD DDEC CONV a|E AD DENCONV a|
TER21 = a(A AD DBSCONV a|B AD BOOCONV a|D AD DDEC CONV a|E AD DENCONV a|
R AD DREECONV a|S AD DSIGCONV a)£ a|N a(A AD DNATCONV £ a|ON£
ASa(£a)AD GENON AR BOOLEXP a)aF
IDENT2 = AR NBSS a|AR EXPQUA aF
IDENT3 = AR NB a|AR EXPQUA aF
IDENT4 = ' AD GEN03 AR CHAINE a|AR EXPQUA aF
IDENT5 = AR NBBOOL a|AR NON a|AR EXPQUA aF
NON = NON£ AD GENON ASa(£a)AR IDENT6 aF
IDENT6 = AR NBBOOL a|AR EXPQUA aF
NBSS = AR NBSS1 ASa(£a) a(E ASa(£a) AR EXPO aJ ERREXP a|aVa) aF
NBSS1 = aI NBSS1 aF
NBBOOL = aI NBBOOL aF
INDTERME = INDEX£ ASa(£a)AD GEN14 a(a(' ASa(a(PADDPREMIER a|DADDDERNIER
a)a|a(AR NBNAT £ aJ ERSEPMOC ASa(£a) A a(P AD DPPREM a|V AD VDERN
a)a)a)aSa(£a) " aJ ERRAPOST a)a|AR EXPQUA a)aF
MOTTERME = MOT£ AD GCONVMOT ASa(£a)a(AR NB a|AR EXPQUA a(AR RAP1 a|aVa)
a)aF
BOOLEXP =AR IDENT6 ASa(£a)a(a(OU AD GENOU2 a|ET AD GENET2 a)£ aJERSEPMOC
ASa(£a)ASa(£a)AR IDENT5 a|aVa)aF
TER4 =_ AD GENMOINS ASa(£a) a(AR ABS ASa(£a)AR IDENT3 ASa(£a)a(AR
ARRONDIA a|aVa) a|a(AR MOD ASa(£a)a('AD GEN03 AR CHAINE a|AR IDENT3
a)a)a|AR IDENT2 a)aF
ABS =A AD DBSCONV £ aF
MOD =M AD DMODCONV £aF
TER6 = AR NBBOOL ASa(£a)a(ET AD GENET1 a|OU AD GENOU1 a)£ aJ ERSEPMOC
ASa(£a)AR IDENT5 aF
TER7 = a(a(ORD £ AD DGENORD1 a|PRE £ AD DGENPRE1 a) ASa(£a) AR EXPQUA
a)a|a(LONG£ AD DGENLON1 ASa(£a) AR IDENT1 ASa(£a)a((AD DRAZNID
ASa(£a) AR PARENT a|aVa)a) aF
RAP1 = a(I AD DINFER a|P AD DPROCHE a|S AD DSUPER a) aF
ARRONDIA = A AD DARRONDI £aSa(£a)a(a(AD RAINE ASa(£a) a(I AD DIEME a|
A AD DAINE a)a)a|U AD DUNITE £aSa(£a) a(AR RAP1 a| aI GENPROCH
a)aF

C

C PHRASES DE TYPE 6 (INSTRUCTIONS)

C

T6 = ASa(AR INSTRUCT a) ASa(AR IDENTDP ASa(£a)a) AD DIRFIN aIFINT6 aF
INSTRUCT = AD DGENT6 ASa(£a)AD NBEEGALO ASa(AR IDENTDP AD NBEPLUS1
ASa(£a)a)a(AR SI a|AR VERS a|AR VIDE a|AR ERREUR a|AR EFFACER a|AR


```

MOUV @|@R
EXPQUA @(:= @D GENAFF @S@(@@)@R TERME @| , ( @D GENAPP @S@(@@)
@R APPEL @|@); @J ERRPTVIR @I FINT6 @F
SI = SI@ @S@(@@)@D GENSI @R COMPAR @S@(@@)@(@R VERS @|@R ERREUR @|@F
VERS = VERS@ @D GENVERS @S@(@@)@(@R FIN @|@R EXPQUA @|@F
VIDE = VIDE @(@@|@D PTVIR @) @D GENVIDE1 @F
ERREUR =ER @D DERREUR @(@@|@D PTVIR @) @D GENERROR @F
EFFACER =EF @D DEFFACER @ @D GENEFFA @S@(@@) @R GEN @F
FIN = FIN @(@@|@D PTVIR @)@D GENVERS1 @F
MOUV = A @ (V@ @D GENINDAV @|R@ @D GENINDAR @)@S@(@@)@R EXPQUA1 @F
IDENTDP = @R IDENT22 @S@(@@): @I GENETIQ @F
IDENT22 = @I IDENT22 @F
IDENT1 = @I IDENT1 @F
APPEL =@D GENNA @(@R ARGD @S@(@@)@D NAPLUS1 @S@ ( , @D NAPLUS1 @S@(@@)
@R ARGD @S@(@@)@|@V@) ; @S@(@@)@D GENNA @(@R EXPQUA @D NAPLUS1
@S@ ( , @D NAPLUS1 @S@(@@)@R EXPQUA @|@|@V@);@S@(@@)@D NAEGALO @ (
@R EXPQUA @D NAPLUS1 @S@ ( ,@D NAPLUS1 @S@(@@)@R EXPQUA @|@|@V@)
) @J ERRPAREN @D GENNA @S@(@@)@S@ ( , @D NAPLUS1 @S@(@@) ( @S@(@@)
@R IDENT1 @D NBEEGAL1 @S@(@@) , @S@(@@)@R EXPQUA @S@ ( , @S@(@@)
@D NBEPUS1 @R EXPQUA @) ) @J ERRPAREN @S@(@@)@|@F
ARGD = @R NB @|@R EXPQUA @F
COMPAR =@R NON2 @S@(@@)@R BOOLEXP @| ' @D GEN50 @R CHAINE @S@(@@)
@R SYMBCOMP @S@(@@) @(@R EXPQUA @|@D TYTPLU10 ' @R CHAINE @)
@| @R EXPQUA @D GENO @(@R SYMCOMP1 @S@(@@)@(@R EXPQUA @|@D
TYTPLU10 @(@R NB @| ' @D TYTPLU10 @R CHAINE @)@)@|@ (OU@ @D GENOU
@|ET@ @D GENET @)@S@(@@)@R IDENT5 @|@F
SYMCOMP1 = @D TRANSFO @R SYMBCOMP @F
NON2 =NON@ @D GENON2 @F
GEN = @ (A @ (D @D DADLIB @|R @D DARTICLE @) @|B @ (L @D DBLANC @|O @D
DBOLEEN @)@|D @D DDECIMAL @|E @D DENTIER @|F @D DFILE @|I @D
DINDEX @|M @D DMOT @|N @D DNATUREL @|O @D DORDRE @|P@ (A @D
DPARTITI @|R @D DPROCEDU @)@|R@ (E @D DREL @|I @D DRIEN @)@)
@ @J ERSEPMOC @S@(@@) @R EXPQUA @F
C
C PHRASE DE TYPE 7 (FIN DU PROGRAMME)
C
T7 = @S@(@@) @DDC7 @R TRAITIM @I FINPASS @F
TRAITIM = @S@(@@) @ ( @D TRAITIM @| @R IDENT1 @) @S@(@@) ; @J ERRPTVIR
@F
C
C

```

REPERTOIRE DES Noms DE REGLES
TAILLE DE L AUTOMATE = 1744

AXIOME	*****	1
10IM	*****	15
11	*****	24
12IM	*****	93
10	*****	19
12	*****	37
13	*****	505
14	*****	1137
15	*****	1168
16	*****	1506
17	*****	1735
IDENT	*****	23
REFE	*****	36
ARG	*****	60
ARGA	*****	101
PER	*****	91
12ENT	*****	270
ARG1	*****	67
12ART	*****	132
12BOL	*****	189
12DEC	*****	251
12FILE	*****	289
12IND	*****	392
12MOT	*****	404
12NAT	*****	416
12ORD	*****	428
12PART	*****	440
12PRBC	*****	463
12REEL	*****	486
12ETIG	*****	110
PLACEPAR	*****	162
PLACE	*****	166
LONG1	*****	201
PLADLIB1	*****	236
LONGNAT	*****	206
EXPR	*****	223
LONGEXPQ	*****	208
EXPQUA2	*****	211
1DENQUA2	*****	939
1DENQUAL	*****	912
NBNAT	*****	222
LONGD	*****	282
LONGD1	*****	286
LONGFILE	*****	320
INDEX	*****	360
EXPRFILE	*****	332
PARENT	*****	346
1DENTR	*****	382
8PER	*****	376
13C	*****	577

I4A	*****	1114
I3A	*****	519
I3012	*****	530
CONDITIO	*****	861
I3020	*****	537
ORAPEAU	*****	544
SUITE1	*****	556
NBFHIS	*****	653
SUITE2	*****	659
CHaine	*****	555
EXPQUA	*****	902
I3E1	*****	581
I3A1	*****	588
I3A2	*****	616
KIENSYM	*****	645
SUITE21	*****	692
SUITE22	*****	701
SUITE23	*****	713
SUITE24	*****	725
SUITE25	*****	743
SUITE26	*****	755
SUITE27	*****	766
SUITE28	*****	778
SUITE29	*****	790
SUITE30	*****	802
SUITE31	*****	814
SUITE32	*****	825
SUITE33	*****	843
SUITE34	*****	849
SYMBCOMP	*****	882
NB	*****	893
NB1	*****	900
EXP0	*****	901
IDENT1	*****	1602
S1	*****	960
EXPQUA1	*****	926
IDENVRAI	*****	955
S21	*****	997
S22	*****	1007
INTERFIL	*****	1074
S24	*****	1028
EXPARFIL	*****	1100
S4	*****	1065
S25	*****	1033
S23	*****	1018
TRANCOD	*****	1045
DEJUSQUA	*****	1088
PREMDERN	*****	1105
VALEUR	*****	1179
VALEUR1	*****	1200
PSEUDO	*****	1210
TERME	*****	1219
IER6	*****	1444

IER1	*****	1246
ARRONDIA	*****	1489
IER4	*****	1420
IER2	*****	1304
IER7	*****	1456
IDENT2	*****	1338
IDENT3	*****	1341
IDENT4	*****	1344
IDENT5	*****	1349
IER21	*****	1314
INDTERRE	*****	1372
MOTTERRE	*****	1397
BOULEXP	*****	1407
NBSS	*****	1364
NBRPDL	*****	1371
NBN	*****	1353
IDENT6	*****	1361
NBSS1	*****	1370
KAP1	*****	1482
ABS	*****	1436
MED	*****	1440
INSTRUCT	*****	1511
IDENTDF	*****	1597
SI	*****	1536
VERS	*****	1547
VIDE	*****	1557
ERREUR	*****	1565
EFFACER	*****	1572
MBOV	*****	1587
APPEL	*****	1603
COMPAR	*****	1654
FIN	*****	1580
GEN	*****	1697
IDENT22	*****	1601
ARGD	*****	1651
NBN2	*****	1691
SYNCOMP1	*****	1688
TRAITIM	*****	1739

REPERTOIRE DES NOMS D'INSV

FINTOIN	*****	1
FINTU	*****	2
IDENTIF	*****	3
FINT1	*****	4
REF	*****	5
FINT2	*****	6
FINT21	*****	7
VERIFPLA	*****	8
VERIFLG	*****	9
NNATUREL	*****	10
FINESSA	*****	11
VERINE	*****	12
FINT3	*****	13
CHAINE	*****	14
MULTI	*****	15
MALTRAI	*****	16
NBBBUL1	*****	17
NB1	*****	18
EXPRESSA	*****	19
FIDENT	*****	20
FINDUGA	*****	21
MAJTYE	*****	22
FINT4A	*****	23
FINT4	*****	24
FINT5	*****	25
NBSS1	*****	26
NBBBUL	*****	27
GENPRBOH	*****	28
FINT6	*****	29
GENE111	*****	30
IDENT22	*****	31
IDENT11	*****	32
FINTPASS	*****	33

REPERTOIRE DES NOMS D'INSF

ERRPTVIR	*****	1
ERR2	*****	2
ERRDECLI	*****	3
ERRSEPAR	*****	4
ERRPTDEC	*****	5
ERRLANG	*****	6
ERRPAREN	*****	7
ERRPVIR	*****	8
ERRPARTI	*****	9
ERRSYNCH	*****	10
ERRNAT	*****	11
ERR7	*****	12
ERREXP	*****	13
IDENSELL	*****	14
ERRSEP	*****	15
ERR8	*****	16
ERRTERME	*****	17
ERREXP	*****	18
ERRAPUST	*****	19

REPERTOIRE DES NOMS DE SEQUENCES DIRECTES

DCO	*****	1
TRAITIM	*****	2
DGENTYP1	*****	3
DAD1	*****	4
DGENT2	*****	5
DDONNEE	*****	6
DMIXTE	*****	7
DRESULTA	*****	8
DAUXILI	*****	9
DAUXILI1	*****	10
DETIGULT	*****	11
DMEMDIR1	*****	12
DARTICLE	*****	13
DGENMEM	*****	14
DGENMEM	*****	15
DCATAL	*****	16
DGENVIDE	*****	17
DDEANDE	*****	18
DCLAVIER	*****	19
DCARTE	*****	20
DERUBAN	*****	21
DIMPRIME	*****	22
DMEMOIRE	*****	23
DPAPIER	*****	24
DCARTIS	*****	25
DSRUBAN	*****	26
DSTOCK	*****	27
DDEOLEEN	*****	28
DGENRIEN	*****	29
DGENAT	*****	30
DGENVAR1	*****	31
GENLGT	*****	32
DGENPLU1	*****	33
DGENVAR3	*****	34
DGENVAR4	*****	35
DDECIMAL	*****	36
DVERIDEC	*****	37
DTIER	*****	38
DFILE	*****	39
VERINDEX	*****	40
DRAZNE	*****	41
DNE1	*****	42
DGENLONG	*****	43
DRAZNID	*****	44
DGENPBIN	*****	45
DNI1	*****	46
DRAZLT	*****	47
DC1	*****	48
GENINDEX	*****	49
D6PERLUE	*****	50
DRAZNI	*****	51

*
* REGLE AXIOME
*

K	TYPE	I	M	V	F
0	DIR	3	DCC	1	0
1	T	0	64	1	2
2	NT	1	TOIM	3	5
3	NT	1	T1	3	4
4	NT	1	T2IM	4	8
5	NT	1	T0	6	0
6	NT	1	T1	6	7
7	NT	1	T2	7	8
8	NT	1	T3	9	0
9	NT	1	T4	9	10
10	NT	1	T5	10	11
11	NT	1	T6	12	0
12	NT	1	T7	13	0
13	FIN	2	AXIOME		

*
* REGLE TOIM
*

K	TYPE	I	M	V	F
14	DIR	3	TRAITIM	15	0
15	T	0	64	15	16
16	T+IF	4	94	17	ERRPTVIR
17	INSV	6	TOIM	FINTOIM	

*
* REGLE T0
*

K	TYPE	I	M	V	F
18	NT	1	IDENT	19	0
19	T	0	64	19	20
20	T+IF	4	94	21	ERRPTVIR
21	INSV	6	T0	FINTO	

*
* REGLE IDENT
*

K	TYPE	I	M	V	F
22	INSV	6	IDENT	IDENTIF	

*
* REGLE T1
*

K	TYPE	I	M	V	F
23	DIR	3	UGENTYP1	24	0
24	T	0	64	24	25
25	NT	1	REFE	26	0
26	DIR	3	DAD1	27	0
27	T	0	64	27	28
28	T	0	107	29	33
29	T	0	64	29	30
30	NT+IF	5	IDENT	31	ERR2

31	DIR	3	DAW1		32	0
32	T	0	64		32	28
33	T+IF	4	94		34	ERRPIVIR
34	INSV	6	11	FINT1		

* REGLE REFE

K	TYPE	I	M	V	F
35	INSV	6	REFE	REF	

* REGLE I2

K	TYPE	I	M	V	F
36	DIR	3	DOENT2	37	0
37	T	0	64	37	38
38	T	0	196	39	43
39	DIR	3	DOENKEE	40	0
40	T	0	64	40	41
41	NT	1	ARG	42	0
42	T	0	64	42	58
43	T	0	212	44	48
44	DIR	3	DMIXTE	45	0
45	T	0	64	45	46
46	NT	1	ARG	47	0
47	T	0	64	47	58
48	T	0	217	49	53
49	DIR	3	DRESULTA	50	0
50	T	0	64	50	51
51	NT	1	ARG	52	0
52	T	0	64	52	58
53	T	0	193	54	0
54	DIR	3	DAUXILI	55	0
55	T	0	64	55	56
56	NT	1	ARGA	57	0
57	T	0	64	57	58
58	INSV	6	I2	FINT2	

* REGLE ARG

K	TYPE	I	M	V	F
59	NT	1	PER	60	0
60	T	0	64	60	61
61	T	0	197	62	64
62	T+IF	4	213	63	ERRDECHI
63	NT	1	I2ENT	65	0
64	NT	1	ARG1	65	0
65	FIN	2	ARG		

* REGLE ARG1

K	TYPE	I	M	V	F
66	T	0	193	67	68
67	NT	1	I2ART	89	0

68	T	0	194	69	70
69	NT	1	T200L	89	0
70	T	0	196	71	72
71	NT	1	T20EC	89	0
72	T	0	198	73	74
73	NT	1	T2FILE	89	0
74	T	0	201	75	76
75	NT	1	T2IND	89	0
76	T	0	212	77	78
77	NT	1	T210T	89	0
78	T	0	213	79	80
79	NT	1	T2NAT	89	0
80	T	0	214	81	82
81	NT	1	T2ORD	89	0
82	T	0	215	83	87
83	T	0	193	84	85
84	NT	1	T2PART	89	0
85	T	0	217	86	0
86	NT	1	T2PRBL	89	0
87	T+IF	4	217	88	ERRDECRI
88	NT	1	T2REEL	89	0
89	FIN	2	ARG1		

*
* REGLE PER

K	TYPE	I	M	V	F
90	T+IF	4	80	91	ERRSEPAR
91	FIN	2	PER		

*
* REGLE T2IM

K	TYPE	I	M	V	F
92	DIR	3	UGENT2	93	0
93	T	0	64	93	94
94	T	0	193	95	0
95	DIR	3	DAUXILI1	96	0
96	T	0	64	96	97
97	NT	1	ARG	98	0
98	T	0	64	98	99
99	INSV	6	T2IM	FINT2IM	

*
* REGLE ARG

K	TYPE	I	M	V	F
100	NT	1	PER	101	0
101	T	0	64	101	102
102	T	0	197	103	107
103	T	0	213	104	105
104	NT	1	T2ENT	108	0
105	T+IF	4	227	106	ERRDECRI
106	NT	1	T2ETIQ	108	0
107	NT	1	ARG1	108	0
108	FIN	2	ARGA		

*** TABLES DES REFERENCES CROISEES ***

* REGLES *

LA REGLE AXIOME N'EST PAS UTILISEE

REGLE T01 UTILISEE 1 FOIS

REGLE T1 UTILISEE 2 FOIS

REGLE T210 UTILISEE 1 FOIS

REGLE T3 UTILISEE 1 FOIS

REGLE T2 UTILISEE 1 FOIS

REGLE T3 UTILISEE 1 FOIS

REGLE T4 UTILISEE 1 FOIS

REGLE T5 UTILISEE 1 FOIS

REGLE T6 UTILISEE 1 FOIS

REGLE T7 UTILISEE 1 FOIS

REGLE IDENT UTILISEE 38 FOIS

18	30	113	135	141	192
254	273	292	300	338	382
488	395	407	419	431	443
449	466	489	508	573	580
697	709	721	739	751	762
774	786	798	810	821	839
857	1145				

REGLE REFF UTILISEE 1 FOIS

REGLE ARG UTILISEE 4 FOIS

41 46 51 97

REGLE ARGA UTILISEE 1 FOIS

REGLE PER UTILISEE 80 FOIS

59	100	111	133	137	139
143	190	194	252	256	271
275	290	294	298	302	306
362	371	393	397	405	409
417	421	429	441	464	487
491	525	539	556	561	566
571	582	602	613	630	642
650	693	695	702	707	714
719	726	737	744	749	756
760	767	772	779	784	791
796	803	815	819	826	837
844	846	850	855	1114	1122
1126	1128	1140	1145	1153	1157
1159	1172				

DC	X'60040100'	0
DC	X'00400001'	4
DC	X'20380103'	8
DC	X'205C0001'	12
DC	X'21700004'	16
DC	X'20480100'	20
DC	X'205C0001'	24
DC	X'20900001'	28
DC	X'27E00100'	32
DC	X'31C00001'	36
DC	X'323C0001'	40
DC	X'37840100'	44
DC	X'3B180100'	48
DC	X'40040000'	52
DC	X'60080100'	56
DC	X'00400001'	60
DC	X'805E0104'	64
DC	X'403C0400'	68
DC	X'20580100'	72
DC	X'00400001'	76
DC	X'805E0104'	80
DC	X'404C0800'	84
DC	X'405C0C00'	88
DC	X'600C0100'	92
DC	X'00400001'	96
DC	X'203C0100'	100
DC	X'60100100'	104
DC	X'00400001'	108
DC	X'006B0105'	112
DC	X'00400001'	116
DC	X'A0580108'	120
DC	X'60100100'	124
DC	X'00400084'	128
DC	X'805E0104'	132
DC	X'40601000'	136
DC	X'40901400'	140
DC	X'60140100'	144
DC	X'00400001'	148
DC	X'00C40105'	152
DC	X'60180100'	156
DC	X'00400001'	160
DC	X'20EC0100'	164
DC	X'00400010'	168
DC	X'00D40105'	172
DC	X'601C0100'	176
DC	X'00400001'	180
DC	X'20EC0100'	184
DC	X'0040000B'	188
DC	X'00D90105'	192
DC	X'60200100'	196
DC	X'00400001'	200

DEUXIEME THESE

Propositions données par la Faculté

« *Organisation d'un ordinateur de gestion* »

Vu
et permis d'imprimer

Le Recteur de l'Académie de Paris

Robert MALLET

Vu et approuvé
Paris, le 16 Avril 1970

Le Doyen de la Faculté des Sciences de Paris

Marc ZAMANSKY